Universidad
Carlos III de Madrid

Universidad Carlos III de Madrid
Escuela Politécnica Superior
Departamento de Informática, Área de Arquitectura de Computadores (ARCOS)

PROYECTO FINAL DE CARRERA DE INGENIERÍA TÉCNICA INFORMÁTICA DE GESTIÓN

---

fsniff: Análisis, diseño e implementación de una suite software para captura y análisis de E/S

Autor:
Javier López Gómez

Tutores:
Alejandro Calderón Mateos
Javier Fernández Muñoz

15 de octubre de 2015

"A la vista de suficientes ojos, todos los errores resultan evidentes."

—The Cathedral and the Bazaar, Eric S. Raymond (ley de Linus)

Índice general

Índice de figuras

Índice de tablas

Resumen

Este proyecto final de carrera ha consistido en la implementación de una suite software para la traza y análisis de acceso a ficheros en el sistema operativo GNU/Linux.

Éste incluye un mecanismo de traza de bajo overhead basado en un módulo de kernel que parchea el kernel Linux que está corriendo. Por tanto, no se requiere recompilación del núcleo ni soporte de instrumentación (Ftrace/Kprobe).

Para el análisis de log, se proveen varias herramientas que permiten extraer streams o la visualización de un grafo de accesos en tiempo real.

Con todo esto se pretende que este proyecto sea de interés para la ingeniería inversa y el perfilaje (puede ser usado para optimizar programas que generan un carga E/S pesada).

Palabras clave— ingeniería inversa, desarrollo kernel, traza E/S, perfilaje E/S

Abstract

This thesis consisted in the implementation of a software suite for file access tracing and analysis on the GNU/Linux operating system.

It includes a low-overhead trace mechanism based on a kernel module that patches the running Linux kernel. Thus, no kernel recompilation nor instrumentation support (Ftrace/Kprobe) is required.

On the log analysis side, several tools that allow stream extraction and real-time access graph visualization are provided.

It is intented that this project might be of interest for reverse engineering and profiling (may be used to optimize programs that generate heavy I/O load).

Keywords— reverse engineering, kernel development, IO trace, IO profiling

Agradecimientos

Este proyecto no habría sido posible sin el apoyo y empuje de mucha gente. Por este motivo desearía dedicar estas líneas a esas personas:

- A mis padres, por su apoyo durante toda la vida.
- A Alejandro Calderón y Javier Fernández, por el interés personal que han tenido en este proyecto y aguantarme durante horas.
- A mi novia, que a pesar de todo el tiempo que he dedicado a esto, me quiere igual.
- A mi primo José Luis, por sus preguntas periódicas del proyecto.
- A mi hermano, por aguantar mis charlas de arquitectura de computadores.
- A David, por interesarse por el estado de esto y su oferta de ayuda.

Sé que me habré dejado a alguien en el tintero y espero que me lo podáis perdonar. Gracias a todos.

Capítulo 1

Introducción

En un sistema operativo monotarea (DOS), si hay accesos a disco cuando no corresponde (e.g. cuando la tarea activa es el interprete de comandos), el sistema está corriendo un TSR (y puede que sea malicioso —virus).

En la actualidad, la situación es diferente: un sistema operativo multitarea y multiusuario permite la ejecución concurrente de varios programas. Éstos pueden acceder a un sistema de ficheros respaldado por un dispositivo de bloques, lo que genera carga E/S.

Debido a que el disco es varios órdenes de magnitud más lento que la CPU, una alta carga E/S se traduce en un rendimiento pobre para algunos procesos, ya que pasarán una parte de su tiempo durmiendo —esperando a que se complete la transferencia.

1.1. Motivación

Como ya se ha comentado, una alta carga E/S se traduce en un rendimiento pobre para aquellos procesos que compiten por acceder a disco[1].

En el capítulo 2 se verá que en GNU/Linux no hay alternativas que puedan usarse para esto del mismo modo que uno usa un sniffer como tcpdump/wireshark (ver figura 1.1) para ver qué está sucediendo en una red.

Si el uso de estas utilidades en el ámbito de las redes permite la solución de problemas y la detección de tráfico sospechoso, aquí puede ser usado para ingeniería inversa, detectar E/S maliciosa o descubrir en qué se consume tiempo[2].

Por este motivo surge este proyecto que pretende proveer herramientas para interceptar operaciones de E/S sin instrumentación y analizar logs generados.

1.2. Problemática

Las alternativas existentes (que se describirán en el capítulo 2) presentan los siguientes problemas:

- Los trazadores de kernel requieren que el kernel haya sido compilado con Ftrace o Kprobe, pero los kernels genéricos incluidos en algunas distribuciones no tienen estas características compiladas. Además, su uso es complicado para un usuario.

[1] En sistemas con poca memoria y swap activado, esto repercute también a procesos en que parte de su espacio de direcciones haya sido paginado.

[2] Lo que un desarrollador puede usar para aplicar optimizaciones.

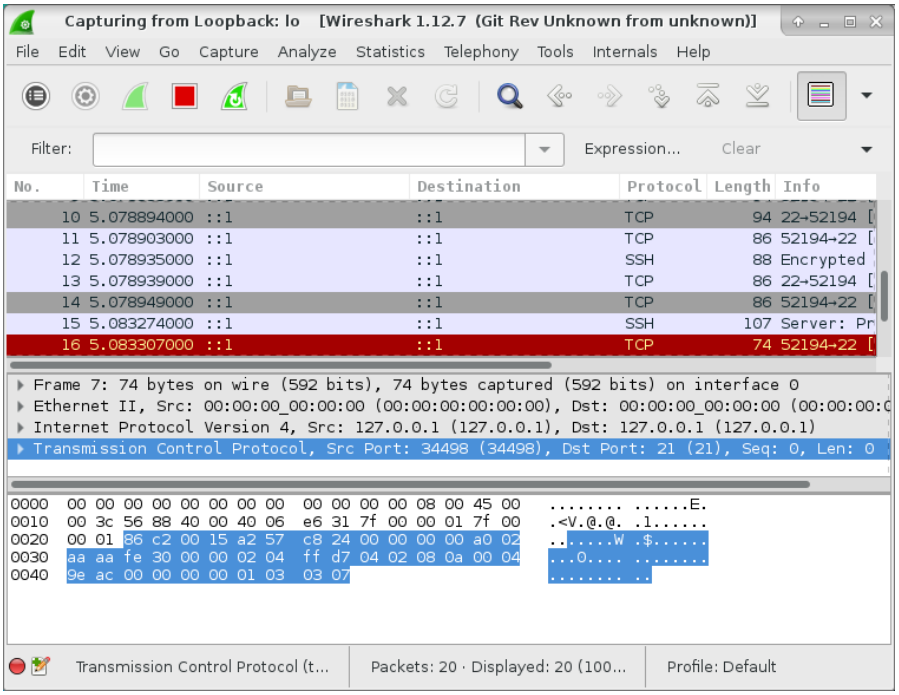


Figura 1.1: Wireshark

- Las alternativas que trazan llamadas a sistema (strace) no pueden interceptar accesos a una región de memoria que haya sido mapeada.

- No incluyen (o incluyen pocas) herramientas para el tratamiento de logs generados. En concreto, no se proveen herramientas para extracción y visualización de datos.

1.3. Objetivos

A continuación se enumeran los objetivos[3] que se pretenden alcanzar con la realización de este proyecto:

Proveer mecanismos de captura sin instrumentación. Los mecanismos de captura provistos no necesitan soporte de instrumentación. Además, pueden interoperan con cualquier herramienta de análisis incluida en el proyecto; para esto se escribirá una capa que permita producir/consumir logs.

Proveer herramientas de análisis. Se incluirán herramientas para extraer streams de un log y visualizar grafos, entre otras.

Aplicar técnicas usadas en rootkits. Trazar desde espacio kernel sin Ftrace implica interceptar el camino normal de ejecución en el kernel usando técnicas que se han considerado típicamente pertenecientes a rootkits como hook de llamadas a sistema o hook de objetos de kernel (KOH).

[3]Su consecución será revisada en el capítulo 9.

Trazas con overhead despreciable. Un módulo de kernel no requiere cambios de contexto o llamadas a sistema. Además, un buffer circular evita la contención productor–consumidor lo que permitirá la traza con overhead despreciable.

Aportar a la comunidad una herramienta GPL para el análisis de E/S. El código fuente del software será liberado bajo la licencia GNU General Public License lo que hará que este software se convierta en la primera solución dedicada a análisis de E/S en GNU/Linux.

Durante el desarrollo emergieron (como parte de los objetivos anteriores) además los siguientes sub-objetivos:

Estudio de los mecanismos de acceso a ficheros (kernel). Estudiar la capa VFS, gestión de memoria, caché de páginas y la capa bIO del kernel Linux para descubrir puntos en los que puede hookearse la ejecución.

Aprendizaje de programación en espacio kernel. Escribir código de kernel añade problemas respecto a la programación en espacio usuario:

- Un fallo de segmentación puede ser fatal para el proceso actual (si no para el sistema)
- Una cuestión de seguridad puede comprometer todo el sistema.
- La depuración con KDB puede ser tediosa.

Aprendizaje GTK+/GDK. Se hará uso de threads GDK, haciendo llamadas GTK sólo desde el hilo principal.

Asegurar escalabilidad. De este modo se conseguirá que la adición de nuevas características pueda hacerse con pocos cambios en el código fuente. Esto reduce los costes de cambios en el código.

Garantizar que el software sea mantenible. Debido a que el software se dejará a la comunidad con licencia GPL, es necesario hacer un buen uso de los comentarios y proveer documentación para nuevos desarrolladores que se unan al proyecto.

1.4. Estructura del documento

En esta sección se incluye una guía de la organización interna de este documento para facilitar la búsqueda a través de éste. El contenido se divide en los capítulos siguientes:

Estado de la cuestión incluye información de otros proyectos emparentados y una comparativa de características de éstos (incluyendo a este proyecto).

Se enumeran también aquellas partes de la bibliografia que debería revisar para comprender el resto del proyecto (arquitectura de computadores y sistemas operativos).

Análisis y diseño incluye requisitos de usuario y casos de uso, especificación de requisitos software (SRS), arquitectura y matrices de trazabilidad.

Implementación este capítulo incluye detalles interesantes de la implementación de varios componentes del proyecto. En algunos casos se incluirá parte del código en el apéndice D en la página 113.

Pruebas y evaluación se enumeran las pruebas que aseguran el funcionamiento y la estabilidad del software y se incluye una comparativa de rendimiento obtenido por los mecanismos de captura provistos bajo diferentes condiciones.

Planificación este capítulo enumera las tareas en las que se dividió el proyecto e incluye un diagrama Gantt en el que se puede apreciar la planificación temporal de éstas.

Presupuesto incluye el presupuesto requerido para llevar a cabo este proyecto.

Trabajos futuros enumera posibles mejoras y trabajos futuros que se derivan de este proyecto.

Conclusiones incluye conclusiones obtenidas de la realización del proyecto y comentarios sobre la liberación del código fuente y de la documentación.

Se recomienda la revisión de los apéndices que incluyen referencia adicional al final de este documento.

Capítulo 2

Estado de la cuestión

En este capítulo se describen otros proyectos que resuelven parcialmente el mismo problema. Como se verá en la sección 2.4, ninguno de ellos soluciona la problemática.

Nota

Si no está familiarizado con conceptos de sistemas operativos se recomienda leer del libro "Understanding the Linux kernel"[DPB06]:

- Capítulo 1 "Introduction": Basic Operating System Concepts
- Capítulo 10 "System calls"
- Capítulo 12 "The Virtual File System"
- Capítulo 16 "Accessing files"

2.1. LTTng

El proyecto LTTng (Linux Trace Toolkit Next Generation) es un paquete para la traza del kernel Linux, aplicaciones y bibliotecas que permite comprender las interacciones entre múltiples componentes de un sistema.

Incluye un módulo de kernel —para trazar el kernel— y objetos compartidos ELF (.so) —para aplicaciones y bibliotecas—.

LTTng necesita soporte de Ftrace o kprobes compilado en el núcleo y está diseñado para alto rendimiento (bajo overhead) —usa técnicas como buffering por-CPU, RCU, un formato binario eficiente y compacto (CTF)....

2.1.1. Características

- Corre con recursos limitados (sistemas empotrados)
- Disponible para varias arquitecturas: x86, ARM, PowerPC, MIPS...
- Soporta múltiples trazas concurrentes

- Permite traza local, remota, en vivo e instantánea
- Incluido en el repositorio de paquetes de las grandes distribuciones
- Herramientas de vista y análisis externas: babeltrace (salida básica de texto que puede ser "grepeado"), Tracecompass (plugin para Eclipse)

2.2. Systemtap

Systemtap es una herramienta que permite a los desarrolladores y administradores de sistemas escribir y reusar scripts sencillos para examinar las actividades un sistema Linux vivo. Los datos pueden ser extraídos, filtrados y resumidos de modo rápido y seguro, para permitir el diagnóstico de problemas de rendimiento y funcionales complejos.

Systemtap funciona traduciendo el script a C y ejecutando el compilador de C para crear un módulo de kernel. Éste módulo es entonces cargado en el kernel. Para que pueda adjuntarse un manejador a un evento se requiere soporte de Ftrace en el núcleo.

Algunas características de fsniff pueden ser fácilmente emuladas con Systemtap (ver listado 2.1 y `http://www.sourceware.org/systemtap/SystemTap_Beginners_Guide/iotimesect.html`.

Listado 2.1: Systemtap: strace_open.stp

```
1  # cat strace_open.stp
2  probe syscall.open
3  {
4    printf("%s(%d) open(%s)\n", execname(), pid(), argstr)
5  }
6  probe timer.ms(4000) # after 4 sec
7  {
8    exit()
9  }
10
11 # stap strace_open.stp
12 vmware-guestd(2206) open("/etc/redhat-release", O_RDONLY)
13 hald(2360) open("/dev/hdc", O_RDONLY|O_EXCL|O_NONBLOCK)
14 hald(2360) open("/dev/hdc", O_RDONLY|O_EXCL|O_NONBLOCK)
15 hald(2360) open("/dev/hdc", O_RDONLY|O_EXCL|O_NONBLOCK)
16 df(3433) open("/etc/ld.so.cache", O_RDONLY)
17 df(3433) open("/lib/tls/libc.so.6", O_RDONLY)
18 df(3433) open("/etc/mtab", O_RDONLY)
19 hald(2360) open("/dev/hdc", O_RDONLY|O_EXCL|O_NONBLOCK)
```

Systemtap está diseñado para gente con conocimiento del kernel intermedio—avanzado. Como consecuencia, es menos útil para administradores o desarrolladores con conocimiento limitado del kernel Linux.

2.2.1. Características

- Simplifica la recolección de información en un Linux que está corriendo
- Elimina la necesidad de instrumentar—recompilar—instalar—reiniciar

- No es tanto una herramienta como es un sistema que permite escribir herramientas forenses y de monitorización para el kernel.

2.3. strace

Strace es una herramienta de diagnóstico, enseñanza y depuración que intercepta las llamadas a sistema invocadas y las señales recibidas por un proceso; el nombre de la llamada a sistema, sus argumentos y el valor devuelto se imprimen por stderr.

Los administradores de sistemas y solucionadores de problemas la usan para resolver problemas en programas para los que el código no está disponible, ya que no necesita ser recompilado para ser trazado.

Strace (igual que fsni-src-pt, parte de este proyecto) usa la llamada a sistema `ptrace()` para interceptar llamadas a sistema.

La salida con las opciones `-e trace=file`, `-e read=set` y `-e write=set` dá una vista general de los ficheros que el proceso está referenciando, pero será imposible trazar accesos a un área de memoria que haya sido mapeado con `mmap()`.

A continuación puede apreciarse parte de la salida generado para `$ strace -e trace=file`:

```
open("/usr/lib/arm-linux-gnueabihf/libgnutls.so.26",
  O_RDONLY) = 15
fstat64(15, {st_mode=S_IFREG|0644, st_size=729260, ...}) = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such
  file or directory)
open("/usr/lib/arm-linux-gnueabihf/libtasn1.so.3", O_RDONLY)
  = 15
fstat64(15, {st_mode=S_IFREG|0644, st_size=54920, ...}) = 0
access("/etc/gcrypt/fips_enabled", F_OK) = -1 ENOENT (No such
  file or directory)
open("/proc/sys/crypto/fips_enabled", O_RDONLY) = -1 ENOENT
  (No such file or directory)
access("/dev/random", R_OK)             = 0
access("/dev/urandom", R_OK)            = 0
open("/dev/urandom", O_RDONLY)          = 15
open("/etc/pkcs11/pkcs11.conf", O_RDONLY) = -1 ENOENT (No such
  file or directory)
open("/etc/pkcs11/modules", O_RDONLY|O_NONBLOCK|O_LARGEFILE
  |O_DIRECTORY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
open("/etc/gnutls/pkcs11.conf", O_RDONLY|O_LARGEFILE) = -1 ENOENT
  (No such file or directory)
open("/etc/ssl/certs/ca-certificates.crt", O_RDONLY|O_LARGEFILE)
  = 16
fstat64(16, {st_mode=S_IFREG|0644, st_size=272519, ...}) = 0
```

2.3.1. Características

- Volcado de todas las llamadas a sistema (útil para solucionar problemas)

Característica	LTTng	Systemtap	strace	fsniff
Soporte S.O.	Linux	Linux	≥4.7 (Linux)	Linux,...
Licencia	LGPL	GPL	GPL	GPL
Arquitecturas	x86, ARM,...	⟨varias⟩	⟨varias⟩	x86, x86_64
Público objetivo	Devs	Devs	Devs, sysadmins	Devs, sysadmins
Necesita instrumentación	Si	Si	No	No
Herramientas de análisis	⟨externo⟩	⟨externo⟩	No	Si
Posible capturar todos los accesos	Si (en teoría)	Si (en teoría)	No	Si
Difícil de aprender	Si	Si	No	No

Tabla 2.1: Matriz de características

- Traza de grupos de syscalls: process, network, signal, ipc, desc.
- Medida de tiempo pasado en una llamada a sistema.

2.4. Comparativa

Como puede apreciarse en la tabla 2.1, ninguna de las soluciones analizadas incluye todas las características.

Capítulo 3

Análisis y diseño

En este capítulo se incluyen requisitos de usuario y casos de uso (secciones 3.2 y 3.3), SRS (sección 3.4), arquitectura del software (3.5).

3.1. Descripción del proyecto

El objetivo (tal cual fue descrito en la sección 1.3) es desarrollar un software que pueda capturar accesos a ficheros originados por un proceso y representar éstos en una forma que resulte útil para el análisis de comportamiento.

Para esto, el software es dividido en dos clases de componentes:

SRC (origen) generan un log para un SINK i.e. implementan los mecanismos necesarios para obtener un log que describa los eventos ocurridos. Serán aquellos programas que el usuario final ejecute para hacer una captura.

SINK (sumidero) éstos proveen al usuario final diferentes representaciones de la información contenida en un log.

El stream de bytes desde un SRC a un SINK será un *libfsni stream*.

3.1.1. Características del usuario

El público objetivo de este proyecto es un ingeniero/técnico con conocimientos avanzados de sistemas UNIX.

Aunque para ejecutar los programas incluídos es suficiente con un nivel medio-bajo en el manejo de estos sistemas, es probable que los resultados obtenidos sólo sean de interés o puedan ser interpretados por una persona con conocimiento avanzado.

Para la ejecución del proyecto se requiere:

- Manejo de un shell UNIX (bash).

3.1.2. Entorno operacional

Este software ha sido probado en máquinas con pocos recursos. El entorno operacional será:

- Arquitectura Intel x86 (ox x86_64), cualquier procesador moderno. Gephi es una aplicación pesada, la máquina que ejecute ésta deberá disponer de más potencia de cómputo y suficiente RAM.
- El medio al que se escriba el log debe tener una alta tasa de transferencia (SSD o HDD 7200rpm).
- Linux kernel >= 4.0.9, FUSE 2.9, GTK+ 3.16.

3.2. Requisitos de usuario

A continuación se describen los requisitos de usuario obtenidos durante la educción de requisitos. Éstos pueden ser de dos tipos:

Requisitos de capacidad Especifican qué debe ser capaz de hacer la solución propuesta.

Requisitos de restricción restricciones sobre los requisitos de capacidad. Éstos pueden ser de interfaz de comunicación, de interfaz hardware, de interfaz software, de interfaz de usuario, de portabilidad, de seguridad o de tiempo.

Los requisitos se especifican usando la plantilla de la figura 3.1. Ésta incluye:

Identificador será XX-UR-YY donde XX puede ser CA(para capacidad) o RE(para restricción), YY es un número de secuencia iniciado en 01. Encabeza la plantilla.

Descripción especificación del requisito con lenguaje no ambiguo.

Necesidad prioridad para el cliente (esencial, conveniente u opcional).

Prioridad prioridad para el desarrollador (alta, media o baja).

Estabilidad especifica si varía durante el desarrollo (no cambia, cambiante o muy inestable).

Verificabilidad del requisito (alta, media o baja).

XX-UR-YY
Descripción:
Necesidad:
Prioridad:
Estabilidad:
Verificabilidad:

Figura 3.1: Plantilla de requisitos de usuario

3.2.1. Requisitos de capacidad

CA-UR-01

Descripción:	Se implementa un SRC basado en FUSE que replica el sistema de ficheros raiz (/) y loguea las operaciones sobre el mismo.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta

CA-UR-02

Descripción:	El sistema puede implementar otros SRC más eficientes, incluyendo un módulo de kernel Linux.
Necesidad:	Conveniente
Prioridad:	Media
Estabilidad:	No cambia
Verificabilidad:	Alta

CA-UR-03

Descripción:	Todos los SRC deben ser capaces de filtrar la captura por path del fichero.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta

CA-UR-04

Descripción:	Una traza incluye una secuencia de eventos. Cada evento incluye información dependiente de evento y opcionalmente una copia del buffer transferido.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta

CA-UR-05

Descripción:	Deben generarse eventos para: ▪ open()/close() para el último *fd* que hace referencia a un fichero. ▪ syscalls que transfieran datos: read()/write() y otras ▪ mmap()/munmap() ▪ lectura/escritura en un mmap
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta

CA-UR-06

Descripción:	El origen de todos los mensajes en una traza debe estar identificado: PID, TGID y comm.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta

CA-UR-07

Descripción:	Los mensajes producidos deben incluir marcas de tiempo de inicio y fin de la operación.
Necesidad:	Conveniente
Prioridad:	Media
Estabilidad:	No cambia
Verificabilidad:	Alta

CA-UR-08

Descripción:	El sistema debe ser capaz de gestionar varias trazas concurrentes.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta

CA-UR-09

Descripción:	Se debe poder dirigir la traza a un proceso o grupo de ellos.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta

CA-UR-10

Descripción:	Al menos un SRC debería ser capaz de capturar toda la E/S generada.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta

CA-UR-11

Descripción:	Debe poderse capturar sin elevación de privilegios, sin más intervención de root que tener el sistema instalado y configurado.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta

CA-UR-12

Descripción:	El usuario debe poder visualizar un volcado de los mensajes generados en una consola de texto.
Necesidad:	Conveniente
Prioridad:	Media
Estabilidad:	No cambia
Verificabilidad:	Alta

CA-UR-13

Descripción:	El usuario debe disponer de una GUI para cargar logs ($\simeq$ wireshark).
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta

CA-UR-14

Descripción:	El usuario debe poder realizar una captura desde la GUI.
Necesidad:	Conveniente
Prioridad:	Media
Estabilidad:	No cambia
Verificabilidad:	Alta

CA-UR-15

Descripción:	Para un fichero F accedido por un proceso P, el usuario puede extraer el stream: ▪ $P \leftarrow F$ (read) ▪ $P \rightarrow F$ (write) ▪ $P \leftrightarrow F$ (ioctl)
Necesidad:	Conveniente
Prioridad:	Media
Estabilidad:	No cambia
Verificabilidad:	Alta

CA-UR-16

Descripción:	Al extraer un stream los fragmentos deben ser escritos en su posición original en el fichero, independiente del orden en que fueron accedidos.
Necesidad:	Conveniente
Prioridad:	Media
Estabilidad:	No cambia
Verificabilidad:	Alta

CA-UR-17

Descripción:	Se debe proveer una lista de ficheros accedidos que incluya el tiempo dedicado a E/S y contadores de bytes transferidos en cada dirección.
Necesidad:	Conveniente
Prioridad:	Media
Estabilidad:	No cambia
Verificabilidad:	Alta

CA-UR-18

Descripción:	Se debe poder generar una definición de grafo en formato DOT que represente la situación para un log.
Necesidad:	Conveniente
Prioridad:	Baja
Estabilidad:	No cambia
Verificabilidad:	Alta

CA-UR-19

Descripción:	El usuario debe poder visualizar desde Gephi en tiempo real qué está ocurriendo en una captura.
Necesidad:	Opcional
Prioridad:	Baja
Estabilidad:	No cambia
Verificabilidad:	Alta

CA-UR-20

Descripción:	El sistema debe poder extenderse para soportar la captura de IOCTLs.
Necesidad:	Opcional
Prioridad:	Baja
Estabilidad:	No cambia
Verificabilidad:	Alta

3.2.2. Requisitos de restricción

Requisitos de interfaz de comunicación

RE-UR-01

Descripción:	Todos los SRC pueden escribir el log a un fichero o a *stdout*.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta

RE-UR-02

Descripción:	Cualquier SRC puede interoperar con cualquier SINK.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta

RE-UR-03

Descripción:	Un log puede incrementar su tamaño sin límite, hasta quedarse sin espacio en el sistema de ficheros.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta

RE-UR-04

Descripción:	El buffer capturado en una lectura/escritura puede contener hasta varios MiB de datos, aunque puede ser truncado por el usuario.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta

RE-UR-05

Descripción:	La interacción con Gephi debe ser compatible con el plugin Graph Streaming.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta

Requisitos de interfaz hardware

RE-UR-06

Descripción:	Un log debería leerse en cualquier arquitectura, incluso si su endianess es diferente.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta

Requisitos de interfaz software

RE-UR-07

Descripción: El proyecto debe componerse de:

- fsni-src-* (SRC).
- fsni-sink-dump (SINK, vuelca mensajes en una consola de texto).
- fsni-sink-gui (SINK, GUI que permite cargar logs/hacer capturas).
- fsni-sink-streams (SINK, extrae streams desde un log).
- fsni-sink-graph (SINK, genera un grafo DOT).
- fsni-sink-gs (SINK, sirve un grafo en tiempo real a Gephi).

Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta

RE-UR-08

Descripción:	La implementación del sistema debe realizarse en C/C++.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta

RE-UR-09

Descripción:	Todos los parámetros se pasan a los programas como argumentos de linea de comando, compatible con getopt (GNU libc).
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta

Requisitos de interfaz de usuario

RE-UR-10

Descripción:	Todos los programas se ejecutan en una consola de texto, a excepción de fsni-sink-gui.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta

Requisitos de portabilidad

RE-UR-11

Descripción:	La solución completa debe poder ejecutarse al menos en las arquitecturas x86 y amd64.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta

RE-UR-12

Descripción:	En las arquitecturas no soportadas deben poder ejecutarse al menos los programas SINK.
Necesidad:	Esencial
Prioridad:	Media
Estabilidad:	No cambia
Verificabilidad:	Alta

RE-UR-13

Descripción:	La solución completa deberá poder ejecutarse en sistemas GNU/-Linux, y en otros sistemas POSIX aunque no estén disponibles todas las características.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta

Requisitos de seguridad

RE-UR-14

Descripción:	Ningún SRC debe comprometer la seguridad de los procesos de otros usuarios.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Media

Requisitos de tiempo

RE-UR-15

Descripción:	El overhead de la escritura de un mensaje en un log debe ser mínimo.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta

RE-UR-16

Descripción:	Se debe proveer un SRC capaz de capturar E/S con una penalización de rendimiento inapreciable.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta

RE-UR-17

Descripción:	La resolución de las marcas de tiempo incluidas en los mensajes debe ser de 1 μs.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta

3.3. Casos de uso

Un diagrama UML de casos de uso es de utilidad durante la educción de requisitos, ya que provee al cliente un vistazo a la funcionalidad del sistema y qué roles externos pueden hacer uso de cada característica.

Un caso de uso describe la secuencia de pasos que un rol externo (actor) debe ejecutar para hacer uso de una funcionalidad.

El diagrama UML de casos de uso se incluye en la figura 3.2.

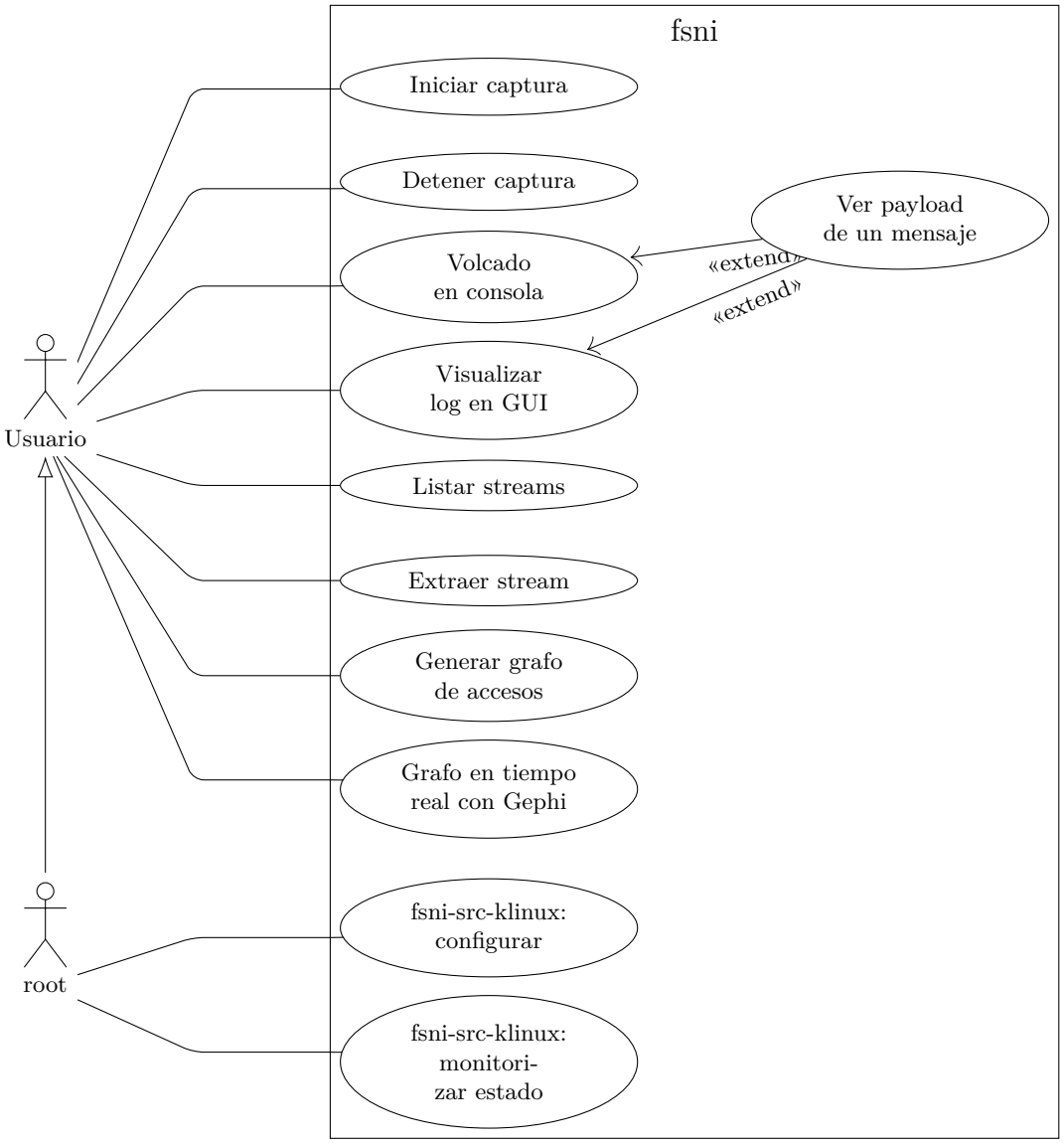


Figura 3.2: Diagrama UML de casos de uso

Los casos de uso se especifican usando la plantilla de la figura 3.3. Ésta describe los aspectos siguientes:

Identificador será UC-XX donde XX es un número de secuencia iniciado en 01. Encabeza la

plantilla.

Nombre descripción corta del caso de uso. Usada en el diagrama de casos de uso de la figura 3.2.

Actores rol externo que ejecuta el caso de uso.

Objetivo del caso de uso.

Descripción especificación en lenguaje natural de la ejecución del caso de uso. Se describen las alternativas más comunes, aunque podría haber otras.

Pre-condición condiciones previas que se verifican antes de la ejecución del caso de uso.

Post-condición condiciones que se verifican tras la ejecución del mismo.

UC-XX

Nombre:

Actores:

Objetivo:

Descripción:

Pre-
condición:

Post-
condición:

Figura 3.3: Plantilla de casos de uso

UC-01

Nombre:	Iniciar captura
Actores:	Usuario
Objetivo:	Iniciar una nueva captura con cualquier SRC.
Descripción:	Desde un terminal: 1. Seleccionar un SRC de fsni-src-fuse, fsni-src-pt o fsni-src-kctl. 2. Establecer objetivo de captura (para fsni-src-pt o fsni-src-kctl). 3. Establecer filtro de captura con las opciones `--allow`/`--deny`. 4. Establecer salida de log con la opción `--output`. 5. Añadir argumentos adicionales. Ejecute sin argumentos o con `--help` para ver qué está disponible. 6. Ejecute la linea de comando. Desde fsni-sink-gui: 1. Click en Capture → New... 2. Escriba el comando de captura como lo haría en un terminal asegurándose de que incluye la opción `--output=-`. 3. Click en OK.
Pre-condición:	El sistema debe estar instalado (ver manual de instalación).
Post-condición:	La captura está activa.

UC-02

Nombre:	Detener captura
Actores:	Usuario
Objetivo:	Detener una captura activa.
Descripción:	Desde un terminal: 1. `kill -SIGINT` o Ctrl+C. Desde fsni-sink-gui: 1. Click en Capture → Stop.
Pre-condición:	Hay una captura activa.
Post-condición:	La captura ha terminado.

UC-03

Nombre:	Volcado en consola
Actores:	Usuario
Objetivo:	Obtener un volcado de mensajes por consola.
Descripción:	Desde un fichero log de una captura previa: 1. `fsni-sink-dump < input.log` Con una tubería desde un SRC: 1. `fsni-src-* [...] | fsni-sink-dump`
Pre-condición:	Existe un log.
Post-condición:	Se obtiene una representación en texto plano de los mensajes.

UC-04

Nombre:	Visualizar log en GUI
Actores:	Usuario
Objetivo:	Obtener un volcado de mensajes desde GUI.
Descripción:	Desde fsni-sink-gui: 1. Click en Capture → Open... 2. Seleccionar fichero log. 3. Click en OK
Pre-condición:	Existe un fichero log de una captura previa.
Post-condición:	El log se visualiza en la GUI.

UC-05

Nombre:	Ver payload de un mensaje
Actores:	Usuario
Objetivo:	Obtener volcado de payload.
Descripción:	Dede un terminal: 1. Incluya la opción `--dump-payload` en la ejecución de fsni-sink-dump. Desde fsni-sink-gui: 1. Click en un mensaje.
Pre-condición:	Existe un log.
Post-condición:	Se obtiene una representación hexadecimal+ASCII del payload.

UC-06

Nombre:	Listar streams
Actores:	Usuario
Objetivo:	Listar ficheros en los que se ha logueado E/S.
Descripción:	Desde un terminal: 1. `fsni-sink-streams -list < input.log`
Pre-condición:	Existe un log.
Post-condición:	Se escribe en *stdout* un listado que incluye path,proceso que accedió y contadores de R/W.

UC-07

Nombre:	Extraer stream
Actores:	Usuario
Objetivo:	Extraer un stream R/W/IOCTL desde un log.
Descripción:	Desde un terminal: 1. `fsni-sink-streams -extract=R|W|IOCTL [-offset= | -path=] <input.log`
Pre-condición:	Existe un log.
Post-condición:	Se escribe en *stdout* el stream solicitado. Si *stdout* es seekable, los fragmentos son escritos en su posición original.

UC-08

Nombre:	Generar un grafo de accesos
Actores:	Usuario
Objetivo:	Generar un grafo dirigido de losaccesos capturados en un log.
Descripción:	Gephi: 1. `fsni-sink-graph < input.log > output.dot` graphviz (neato): 1. `fsni-sink-graph | neato -Tpng:cairo:gd < input.log > output.png`
Pre-condición:	Existe un log.
Post-condición:	Se escribe en *stdout* elgrafo en formato DOT.

UC-09

Nombre:	Grafo en tiempo real con Gephi
Actores:	Usuario
Objetivo:	Visualizar un grafo de accesosen tiempo real.
Descripción:	Ejecute el servidor fsni-sink-gs: 1. `fsni-src-* [...] | fsni-sink-gs -bind-addr 127.0.0.1 -bind-port 2480` Desde Gephi: 1. Streaming tab → click en + 2. Source URL: `http://localhost:2480/` 3. OK
Pre-condición:	Existe un log.
Post-condición:	Gephi actualiza un grafo de accesos en tiempo real.

UC-10

Nombre:	fsni-src-klinux: configurar
Actores:	root
Objetivo:	Establecer parámetros de módulo dekernel Linux.
Descripción:	El módulo no está cargado: 1. `modprobe fsni-src-klinux [param=value[,...]]` El módulo está cargado: 1. `echo -n value > /sys/module/fsni-src-klinux/parameters/param` Consulte el manual de instalación para ver los parámetros disponibles.
Pre-condición:	El sistema debe estar instalado (ver manual de instalación). El módulo puede estar o no cargado.
Post-condición:	Se han establecido los parámetros. Si el módulo no estaba cargado, se habrá cargado.

UC-11

Nombre:	fsni-src-klinux: monitorizar estado
Actores:	root
Objetivo:	Monitorizar estado del módulode kernel Linux (consulte el manual de instalación).
Descripción:	Desde un terminal: 1. `cat /proc/fsni-src-klinux`
Pre-condición:	El módulo está cargado.
Post-condición:	Se escribe en *stdout* el estado del módulo.

3.4. Requisitos de software

En esta sección se incluye la especificación de requisitos de software (SRS). Éstos han sido derivados de los requisitos de usuario de la sección 3.2 y describen de forma rigurosa y no ambigua lo que el analista ha entendido de aquellos.

Los requisitos de software pueden ser de dos tipos:

Requisitos funcionales Especifican características funcionales del software.

Requisitos no funcionales Especificaciones adicionales que no añaden nuevas funciones. Éstos pueden ser de interfaz de comunicación, de interfaz hardware, de interfaz software, de interfaz de usuario, de portabilidad, de seguridad o de tiempo.

Los requisitos de software se especifican usando la plantilla de la figura 3.4. Ésta es similar a la plantilla de la figura 3.1 en la página 10, a excepción de:

Identificador Encabeza la plantilla. Será XX-YY-ZZ donde

- XX puede ser F (para requisitos funcionales) o NF (para requisitos no funcionales)
- YY será SR (para requisitos funcionales), IC (para requisitos de interfaz de comunicación), IH (para requisitos de interfaz hardware), IS (para requisitos de interfaz software), IU (para requisitos de interfaz de usuario), PO (para requisitos de portabilidad), SE (para requisitos de seguridad) o TM (para requisitos de tiempo)
- ZZ es un número de secuencia iniciado en 01

Origen referencia a los requisitos de usuario que originaron este requisito de software (ver sección 3.2).

XX-YY-ZZ

Descripción:

Necesidad:

Prioridad:

Estabilidad:

Verificabilidad:

Origen:

Figura 3.4: Plantilla de requisitos software

3.4.1. Requisitos funcionales

F-SR-01

Descripción:	Se implementará un SRC basado en un sistema de ficheros FUSE de dos capas: **log** loguea operaciones sobre el sistema de ficheros. **mirror** los métodos `fuse_operations` implementados redirigen las operaciones a /.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	CA-UR-01, CA-UR-10

F-SR-02

Descripción:	Se implementa un SRC basado en ptrace() para interceptar llamadas a sistema ejecutadas por un proceso.
Necesidad:	Conveniente
Prioridad:	Media
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	CA-UR-02, CA-UR-09

F-SR-03

Descripción:	El SRC basado en ptrace() intercepta las llamadas a sistema: ▪ open, openat, creat, close: referencia a un fichero ▪ lseek, _llseek: alterar puntero de lectura/escritura secuencial ▪ dup, dup2, dup3, fcntl, fcntl64: duplicación de descriptor de fichero ▪ mmap, mmap2, munmap: mapeo/desmapeo en memoria ▪ read, write, pread64, pwrite64: lectura/escritura ▪ readv, writev, preadv, pwritev: lectura/escritura vectorizada ▪ io_submit: E/S asíncrona ▪ ioctl
Necesidad:	Conveniente
Prioridad:	Media
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	CA-UR-02

F-SR-04

Descripción:	Se implementa un SRC basado en un módulo de kernel Linux que acepta solicitudes de un proceso y exporta el log a *userland* via un dispositivo de caracteres.
Necesidad:	Conveniente
Prioridad:	Media
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	CA-UR-02, CA-UR-10

F-SR-05

Descripción:	La solicitud al módulo de kernel puede ser para trazar ▪ a un PID o comm especificado ▪ a todos los procesos con mismo UID (no restringido para root).
Necesidad:	Conveniente
Prioridad:	Media
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	CA-UR-02, CA-UR-09

F-SR-06

Descripción:	El módulo de kernel intercepta un fichero antes de retornar de una llamada a sistema `open()`, `openat()` o `creat()`.
Necesidad:	Conveniente
Prioridad:	Media
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	CA-UR-02

F-SR-07

Descripción:	Se incluye una implementación de filtro de captura basado en path que será usada por todos los SRC.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	CA-UR-03

F-SR-08

Descripción:	La implementación de filtro de captura hace uso de: ▪ Lista de reglas que especifican si un path se permite/-bloquea. ▪ Política por default.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	CA-UR-03

F-SR-09

Descripción:	El módulo de kernel permite monitorizar su estado usando *procfs*.
Necesidad:	Conveniente
Prioridad:	Media
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	CA-UR-02

F-SR-10

Descripción:	Se implementa una librería (libfsni) para generar/leer un log que será usada por todos los SRC/-SINK. La especificación completa puede encontrarse en el apéndice A en la página 93.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	CA-UR-04, CA-UR-05, CA-UR-06, CA-UR-07

F-SR-11

Descripción:	El formato de un log será
	`+-----+-----+-----+-----+`
	`| HDR | MSG | MSG | ... |`
	`+-----+-----+-----+-----+`
	donde HDR es la cabecera del log (ver F-SR-12) y MSG es un mensaje (ver F-SR-13).

Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	CA-UR-04

F-SR-12

Descripción: La cabecera de un log incluirá:

magic_number identifica un log válido; detección del orden de bytes

major_version usado para señalar incompatibilidad con otras versiones de libfsni.

minor_version revisión

payload_max tamaño máximo de un payload

page_size tamaño de página de la máquina

generator SRC generador de este log

Necesidad: Esencial

Prioridad: Alta

Estabilidad: No cambia

Verificabilidad: Alta

Origen: CA-UR-04

F-SR-13

Descripción: Un mensaje incluirá:

type tipo del mensaje (ver F-SR-14)

flags ver documentación en la página 57

error *errno* si hubo un error en la operación; no usado en otro caso

seq número de secuencia

length tamaño total del mensaje en bytes

source origen del mensaje i.e. PID, TGID, comm y marcas de tiempo

msg_data datos dependientes del mensaje

payload carga del mensaje; usado para incluir buffers

Necesidad: Esencial

Prioridad: Alta

Estabilidad: No cambia

Verificabilidad: Alta

Origen: CA-UR-04, CA-UR-06, CA-UR-07

F-SR-14

Descripción: libfsni puede manejar los tipos de mensaje siguientes –consulte la documentación completa en la página 93.

- OPEN
- RELEASE
- RW
- RW_ITER
- IOVEC
- IOCTL
- IOCTL_DATA
- MMAP
- MUNMAP
- VM_FAULT
- WRITEPAGE

Necesidad: Esencial

Prioridad: Alta

Estabilidad: No cambia

Verificabilidad: Alta

Origen: CA-UR-05

F-SR-15

Descripción: El módulo de kernel Linux aceptará solicitudes hasta alcanzar el máximo que fue establecido por root.

Necesidad: Esencial

Prioridad: Alta

Estabilidad: No cambia

Verificabilidad: Alta

Origen: CA-UR-08

F-SR-16

Descripción: El módulo de kernel implementa grupos multicast –entrega de un mensaje producido por un publicador a grupo de suscriptores donde

Publicador es un objeto `file`, `vm_area_struct` o `page`

Suscriptor cualquier solicitud de traza

Necesidad: Conveniente

Prioridad: Media

Estabilidad: No cambia

Verificabilidad: Alta

Origen: CA-UR-02, CA-UR-08, CA-UR-10

F-SR-17

Descripción:	Se incluye una regla *udev* que establecerá los permisos para que el dispositivo de caracteres pueda ser usado por cualquier usuario.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	CA-UR-11

F-SR-18

Descripción:	El módulo de kernel integra un algoritmo para descubrir la dirección de los símbolos `sys_call_table` y `ia32_sys_call_table` en las arquitecturas x86 y amd64 sin intervención de root.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	CA-UR-11

F-SR-19

Descripción:	fsni-sink-dump escribe en *stdout* una representación en texto plano de un log provisto en *stdin*.
Necesidad:	Conveniente
Prioridad:	Media
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	CA-UR-12

F-SR-20

Descripción:	fsni-sink-gui será implementado en GTK+-3 y permitirá la carga de logs en un widget GtkTreeView.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	CA-UR-13

F-SR-21

Descripción:	Cuando las llamadas a sistema tee()/splice() estén disponibles, fsni-sink-gui permite ejecutar un SRC y refrescar el GtkTreeView en tiempo real.
Necesidad:	Conveniente
Prioridad:	Media
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	CA-UR-14

F-SR-22

Descripción:	Referido a F-SR-21; *stdout* del SRC será volcado a un fichero en /tmp para permitir la carga de payload bajo demanda.
Necesidad:	Conveniente
Prioridad:	Media
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	CA-UR-14

F-SR-23

Descripción:	Referido a F-SR-21; *stderr* del SRC será redirigido a una ventana de errores.
Necesidad:	Conveniente
Prioridad:	Media
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	CA-UR-14

F-SR-24

Descripción:	fsni-sink-gui permitirá parar la captura, terminando el proceso SRC ejecutado.
Necesidad:	Conveniente
Prioridad:	Media
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	CA-UR-14

F-SR-25

Descripción:	fsni-sink-gui permitirá guardar la captura a un fichero que será especificado por el usuario.
Necesidad:	Conveniente
Prioridad:	Media
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	CA-UR-14

F-SR-26

Descripción:	Mientras haya un SRC en ejecución, la GUI actualiza contadores de ▪ bytes que han pasado por la tubería ▪ pérdidas de mensajes
Necesidad:	Conveniente
Prioridad:	Media
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	CA-UR-14

F-SR-27

Descripción:	El programa fsni-sink-streams escribirá en *stdout* el stream formado por los payloads de mensajes del tipo RW o IOCTL_DATA que referencian el fichero especificado. Si *stdout* es seekable, cada fragmento se escribe en la posición incluida en datos dependientes de mensaje.
Necesidad:	Conveniente
Prioridad:	Media
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	CA-UR-15, CA-UR-16

F-SR-28

Descripción:	fsni-sink-streams pemitirá referenciar un fichero por ▪ offset (ver F-SR-29) ▪ path
Necesidad:	Conveniente
Prioridad:	Media
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	CA-UR-15

F-SR-29

Descripción:	Ejecutar fsni-sink-streams sin argumentos escribe por *stdout* la lista de ficheros que fueron accedidos. Ésta incluye: ▪ offset ▪ bytes leídos/escritos ▪ tiempo E/S ▪ TGID, PID, comm ▪ path
Necesidad:	Conveniente
Prioridad:	Media
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	CA-UR-17

F-SR-30

Descripción:	El módulo de kernel exporta símbolos que permiten a otros módulos registrar funciones que serán invocadas desde el hook `unlocked_ioctl()`.
Necesidad:	Opcional
Prioridad:	Baja
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	CA-UR-02, CA-UR-20

F-SR-31

Descripción:	Se incluye un módulo que demuestra cómo se puede extender el hook `unlocked_ioctl()`. Éste captura el ioctl `SNDRV_PCM_IOCTL_WRITEI_FRAMES` de ALSA.
Necesidad:	Opcional
Prioridad:	Baja
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	CA-UR-02, CA-UR-20

F-SR-32

Descripción:	Para F-SR-33–F-SR-35, se especifica: **Nodo** podría ser de Tipo-0 (proceso) o Tipo-1 (fichero) **Arista** conecta un nodo Tipo-0 con uno Tipo-1 si ha habido un acceso **Dirección** es dependiente de la dirección de la transferencia
Necesidad:	Conveniente
Prioridad:	Baja
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	CA-UR-18, CA-UR-19

F-SR-33

Descripción:	fsni-sink-graph generará un grafo dirigido en formato DOT desde un log provisto en *stdin*.
Necesidad:	Conveniente
Prioridad:	Baja
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	CA-UR-18

F-SR-34

Descripción:	fsni-sink-gs implementa un servidor al que Graph Streaming (Gephi) podrá conectarse para recibir actualizaciones del grafo.
Necesidad:	Opcional
Prioridad:	Baja
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	CA-UR-19

F-SR-35

Descripción:	fsni-sink-gs envía periódicamente una actualización de atributo para los nodos tipo fichero **b** bytes transferidos
Necesidad:	Opcional
Prioridad:	Baja
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	CA-UR-19

3.4.2. Requisitos no funcionales

Requisitos de interfaz de comunicación

NF-IC-01

Descripción:	Todo SRC incluye la opción `--output` para especificar el path del log. Si se especifica `--output -` la salida será a *stdout*.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	RE-UR-01

NF-IC-02

Descripción:	Todo SRC/SINK depende de libfsni para producir/consumir un log.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	RE-UR-02

NF-IC-03

Descripción:	libfsni invoca funciones de bajo nivel para escribir un mensaje que deberán ser implementadas por el SRC.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	RE-UR-02

NF-IC-04

Descripción:	libfsni mantiene sincronizado un SINK con un SRC usando un número de secuencia.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	RE-UR-03

NF-IC-05

Descripción:	El número de secuencia de libfsni es un espacio circular de 32-bit i.e. $0 < 0xFFFFFFFF < 0$. Un número de secuencia fuera de orden es tratado por libfsni: ▪ contabilizar secuencias perdidas ▪ resincronizar consumidor
Necesidad:	
Prioridad:	
Estabilidad:	
Verificabilidad:	
Origen:	RE-UR-03

NF-IC-06

Descripción:	El tamaño total de un mensaje se incluye en el miembro `length` de la cabecera, que es un entero sin signo de 32-bit. El tamaño máximo para un payload será $\simeq 4GiB$.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	RE-UR-04

NF-IC-07

Descripción:	libfsni puede no devolver el payload al usuario si el tamaño de éste es >`payload_max` especificado en la cabecera.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	RE-UR-04

NF-IC-08

Descripción:	La comunicación entre fsni-sink-gs y Gephi usa JSON/HTTP compatible con el plugin Graph Streaming.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	RE-UR-05

Requisitos de interfaz hardware

NF-IH-01

Descripción:	libfsni escribe mensajes usando el *endianess* (orden de bytes) nativo de la arquitectura.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	RE-UR-06, RE-UR-15

NF-IH-02

Descripción:	El miembro `magic_number` de la cabecera log permite detectar el orden de bytes i.e. la arquitectura interpretará 0xDEADBEEF como 0xEFBEADDE.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	RE-UR-06

NF-IH-03

Descripción:	Si el orden de bytes $\neq$ nativo, libfsni intercambiará los bytes antes de retornar una estructura.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	RE-UR-06

Requisitos de interfaz software

NF-IS-01

Descripción:	fsni-src-fuse implementa los métodos `fuse_operations` aplicables, excepto ioctl().
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	RE-UR-07

NF-IS-02

Descripción:	fsni-src-klinux define los parámetros siguientes: **sys__call__table** especificación manual de la dirección de este símbolo **allow__max** permitir hasta... trazas concurrentes **trace__nonregular** si se trazarán ficheros no regulares **bufsize__max** tamaño máximo para un buffer circular **bufsize__min** tamaño mínimo (por default) **null__writepages** si el método `writepages` de `address_space_operations` se cambiará a NULL.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	RE-UR-07

NF-IS-03

Descripción:	fsni-src-fuse usará la versión 30 de la API FUSE.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	RE-UR-07

NF-IS-04

Descripción:	fsni-src-klinux: la interfaz entre kernel y *userland* es un dispositivo de caracteres.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	RE-UR-07

NF-IS-05

Descripción:	fsni-src-klinux: el dispositivo de caracteres acepta los ioctl siguientes: ▪ IOC_TRACEREQUEST ▪ IOC_TR_ENABLE ▪ IOC_TR_ACCEPT_DELIVERY ▪ IOC_TRACESTAT
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	RE-UR-07

NF-IS-06

Descripción:	La llamada a sistema `read()` en el dispositivo de caracteres consumirá el log desde el buffer circular.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	RE-UR-07

NF-IS-07

Descripción:	La llamada a sistema `close()` en el dispositivo de caracteres cancelará la traza y liberará la memoria en uso.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	RE-UR-07

NF-IS-08

Descripción:	El sistema se implementará en ISO C99 (GNU) y C++11 compilable con gcc 4.6.3.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	RE-UR-08

NF-IS-09

Descripción:	Las lineas de comando serán parseadas con `getopt_long()` de glibc.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	RE-UR-09

Requisitos de interfaz de usuario

NF-IU-01

Descripción:	Los programas pueden ser interrumpidos enviando una señal POSIX `SIGINT` (Ctrl+C).
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	RE-UR-10

NF-IU-02

Descripción:	fsni-sink-gui construye la interfaz con GtkBuilder usando un XML generado con Glade.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	RE-UR-10

Requisitos de portabilidad

NF-PO-01

Descripción:	fsni-src-fuse compila y ejecuta en cualquier arquitectura y sistema POSIX al que el proyecto FUSE haya sido portado.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	RE-UR-11, RE-UR-13

NF-PO-02

Descripción:	fsni-src-pt compila y ejecuta en Linux x86 y amd64.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	RE-UR-11, RE-UR-13

NF-PO-03

Descripción:	fsni-src-klinux puede compilarse en linux >= 4.0.9 (2.6 con algunos cambios) y puede ejecutarse en cualquier arquitectura si `sys_call_table` es especificado manualmente.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	RE-UR-11, RE-UR-13

NF-PO-04

Descripción:	El resto de componentes compilarán y ejecutarán en cualquier combinación arquitectura—sistema POSIX.
Necesidad:	Esencial
Prioridad:	Media
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	RE-UR-12

NF-PO-05

Descripción:	El kernel no permite la carga de un módulo si no coincide su ABI/versiones de símbolos.
Necesidad:	Esencial
Prioridad:	Media
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	RE-UR-11

Requisitos de seguridad

NF-SE-01

Descripción:	Notas de seguridad para fsni-src-fuse: ▪ No puede ser ejecutado como root. ▪ La opción `fuse.conf: user_allow_other` no se incluye por default.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Media
Origen:	RE-UR-14

NF-SE-02

Descripción:	Notas de seguridad para fsni-src-klinux: sólo se interceptarán procesos del mismo UID que solicitó la traza (excepto root).
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Media
Origen:	RE-UR-14

NF-SE-03

Descripción:	Notas de seguridad para fsni-src-pt: la política se seguridad es forzada por el kernel.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Media
Origen:	RE-UR-14

Requisitos de tiempo

NF-TM-01

Descripción:	fsni-src-fuse se ejecutará por default usando FUSE multihilo.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	RE-UR-16

NF-TM-02

Descripción:	fsni-src-fuse implementa los métodos `read_buf()` y `write_buf()` (passthrough) que serán usados si no se pasa la opcion `--with-buffer`.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	RE-UR-15, RE-UR-16

NF-TM-03

Descripción:	fsni-src-klinux reduce la contención usando un buffer circular que evita primitivas de sincronización entre el productor y el consumidor.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	RE-UR-15, RE-UR-16

NF-TM-04

Descripción:	libfsni usa `struct timeval` i.e. la resolución es de 1μs.
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	No cambia
Verificabilidad:	Alta
Origen:	RE-UR-17

3.5. Arquitectura

Esta sección describe la aquitectura de capas del software (ver figura 3.6). Los componentes se especifican usando la plantilla de la figura 3.5 que especifica:

Identificador Nombre interno del componente. Encabeza la plantilla.

Rol del componente en el proyecto.

Dependencias componentes que dependen de éste en tiempo de compilación o ejecución.

Descripción corta del comportamiento de éste.

Datos entradas[in]/salidas[out] de éste.

Recursos usados por el componente.

Origen requisitos software que lo originaron (ver sección 3.4).

Identificador

Rol:

Dependencias:

Descripción:

Datos: n/a

Recursos: n/a

Origen:

Figura 3.5: Plantilla de componentes

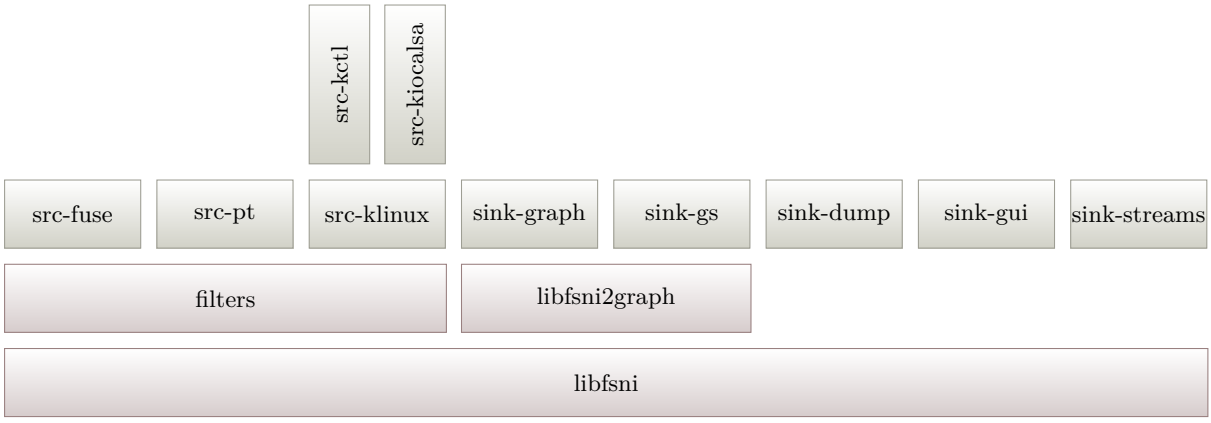


Figura 3.6: Arquitectura de capas del proyecto

3.5.1. Componentes incluidos

libfsni

Rol:	Producción/consumo de logs
Dependencias:	fsni-src-fuse, fsni-src-pt, fsni-src-klinux, fsni-src-kiocalsa,fsni-sink-dump, fsni-sink-gui, fsni-sink-streams, fsni-sink-graph,fsni-sink-gs
Descripción:	Exporta símbolos que pueden ser usados por otros componentes: ▪ Tipos para representar mensajes ▪ API productor/consumidor ▪ Funciones de soporte: hex-dump... ▪ El productor debe implementar cómo se escribe una secuencia de bytes
Datos:	▪ [in] argumentos pasados ▪ [in out] libfsni stream
Recursos:	▪ glibc (sólo consumidor)
Origen:	F-SR-10, F-SR-11, F-SR-12, F-SR-13, F-SR-14, NF-IC-03, NF-IC-04, NF-IC-05,NF-IC-06, NF-IC-07, NF-IS-08, NF-IH-01, NF-IH-02, NF-IH-03, NF-TM-04

filters

Rol:	Manejo de conjuntos de reglas para la captura
Dependencias:	fsni-src-fuse, fsni-src-pt, fsni-src-k
Descripción:	Exporta funciones que pueden ser usadas por los SRC para filtrar la captura por path.
Datos:	▪ [in] argumentos pasados
Recursos:	▪ glibc (para userspace) ▪ kalloc/kfree (para kernel)
Origen:	F-SR-07, F-SR-08, NF-IS-08

fsni-src-fuse

Rol:	Captura usando un sistema de ficheros FUSE
Dependencias:	
Descripción:	1. Se monta un sistema de ficheros FUSE; libfuse espera mensajes del módulo fuse 2. libfuse recibe un mensaje del módulo de kernel fuse e invoca la operación `fuse_operations` 3. Cada path referenciado se traduce añadiendo el preámbulo especificado en la opción `--mirror` 4. Se redirige la operación usando los paths traducidos
Datos:	▪ [in] argumentos linea de comando ▪ [out] libfsni stream
Recursos:	▪ fuse/libfuse ▪ glibc
Origen:	F-SR-01, NF-IC-01, NF-IC-02, NF-IS-01, NF-IS-03, NF-IS-08, NF-IS-09,NF-IU-01, NF-PO-01, NF-SE-01, NF-TM-01, NF-TM-02

fsni-src-pt

Rol:	Captura usando ptrace()
Dependencias:	
Descripción:	1. Se invoca `fork()`, `ptrace(PTRACE_TRACEME)` y `execve()` 2. El proceso hijo se para cuando invoca una llamada a sistema; inspección del número de syscall 3. Algunas llamadas a sistema requieren invocar a `ptrace(PTRACE_PEEKDATA)` para copiar memoria del proceso hijo 4. Éste es parado al retornar a userspace; inspección del valor de retorno
Datos:	▪ [in] argumentos linea de comando ▪ [out] libfsni stream
Recursos:	▪ glibc ▪ libstdc++
Origen:	F-SR-02, F-SR-03, NF-IC-01, NF-IC-02, NF-IS-08, NF-IS-09,NF-IU-01, NF-PO-02, NF-SE-03

fsni-src-klinux

Rol:	Captura usando un módulo de kernel Linux
Dependencias:	fsni-src-kiocalsa, fsni-src-kctl
Descripción:	1. Se hookean las llamadas a sistema `open()`, `openat()` y `creat()`. 2. Cuando una de éstas es invocada, se genera la lista de suscriptores para este fichero; si hay algún suscriptor, se parchea la tabla `file_operations`. 3. La función `mmap` de `file_operations` parchea la tabla `vm_operations_struct` antes de retornar. 4. La función `page_mkwrite` de `vm_operations_struct` parchea la tabla `address_space_operations`; la función `writepage` de ésta será invocada cuando el kernel vuelque una página a disco. Vea el capítulo 4 para más información de éste componente.
Datos:	▪ [in] solicitud desde fsni-src-kctl ▪ [out] libfsni stream
Recursos:	▪ Linux kernel >=4.0.9
Origen:	F-SR-04, F-SR-05, F-SR-06, F-SR-09, F-SR-15, F-SR-16, F-SR-17,F-SR-18, F-SR-30, NF-IC-02, NF-IS-02, NF-IS-04, NF-IS-05, NF-IS-06,NF-IS-07, NF-IS-08, NF-PO-03, NF-PO-05, NF-SE-02, NF-TM-03

fsni-src-kiocalsa

Rol:	Extiende fsni-src-klinux para generar IOCTL_DATApara el ioctl `SNDRV_PCM_IOCTL_WRITEI_FRAMES`
Dependencias:	
Descripción:	1. Registra un función para ser invocada desde el hook `unlocked_ioctl`. 2. Ésta entrega mensajes IOCTL_DATA a todos los sucriptores copiando los frames como payload.
Datos:	▪ [in] argumentos ioctl() ▪ [out] IOCTL_DATA
Recursos:	▪ fsni-src-klinux ▪ Linux kernel
Origen:	F-SR-31, NF-IC-02, NF-IS-08, NF-PO-05

fsni-src-kctl

Rol:	Control de fsni-src-klinux desde espacio usuario
Dependencias:	
Descripción:	1. `open("/dev/fsni-src-klinux")` 2. Invoca `ioctl()` para establecer los parámetros de la traza y habilitarla. 3. Leer desde el dispositivo de caracteres; escribir a otro descriptor de fichero (`sendfile()`).
Datos:	▪ [in] argumentos linea de comando ▪ [out] libfsni stream
Recursos:	▪ glibc ▪ fsni-src-klinux
Origen:	F-SR-04, NF-IC-01, NF-IS-04, NF-IS-05, NF-IS-06, NF-IS-07, NF-IS-08,NF-IS-09, NF-IU-01

libfsni2graph

Rol:	Mantener un grafo en memoria desde un libfsni stream
Dependencias:	fsni-sink-graph,fsni-sink-gs
Descripción:	Consume mensajes manteniendo el estado en dos `std::map` (nodos y aristas)
Datos:	▪ [in] libfsni stream
Recursos:	▪ libstdc++
Origen:	F-SR-32, NF-IS-08, NF-PO-04

fsni-sink-dump

Rol:	Vuelca un log en texto plano.
Dependencias:	
Descripción:	1. libfsni es usado para leer un log. 2. Cada mensaje devuelto se vuelca en texto plano (puede incluir un volcado hexadecimal+ASCII del payload).
Datos:	▪ [in] libfsni stream
Recursos:	▪ libfsni ▪ glibc
Origen:	F-SR-19, NF-IC-02, NF-IS-08, NF-IS-09, NF-IU-01, NF-PO-04

fsni-sink-gui

Rol:	Cargar un log/hacer una captura (GUI)
Dependencias:	
Descripción:	1. GtkBuilder instancia los objetos descritos en una *GtkBuilder UI definition* (XML). 2. Autoconecta algunas señales a funciones de `main.c`. Ver el capítulo 4 para más información.
Datos:	▪ [in] libfsni stream
Recursos:	▪ glibc ▪ X11 (GDK) ▪ GTK>=3.16 ▪ libfsni
Origen:	F-SR-20, F-SR-21, F-SR-22, F-SR-23, F-SR-24, F-SR-25, F-SR-26,NF-IC-02, NF-IS-08, NF-IU-02, NF-PO-04

fsni-sink-streams

Rol:	Operaciones con streams contenidos en un log
Dependencias:	
Descripción:	1. Usa libfsni para consumir mensajes y generar una lista de ficheros abiertos. 2. Si se pasa `-extract`: localiza el principio de un stream y escribe payloads a stdout hasta encontrar un mensaje RELEASE.
Datos:	▪ [in] libfsni stream
Recursos:	▪ libstdc++
Origen:	F-SR-27, F-SR-28, F-SR-29, NF-IC-02, NF-IS-08, NF-IS-09, NF-PO-04

fsni-sink-graph

Rol:	Generar un digrafo desde un log
Dependencias:	
Descripción:	1. Usa libfsni2graph para obtener el conjunto de nodos y aristas que representan el estado para un log. 2. Este estado se vuelca en lenguaje DOT
Datos:	▪ [in] libfsni stream ▪ [out] DOT/GEXF
Recursos:	▪ libstdc++
Origen:	F-SR-32, F-SR-33, NF-IC-02, NF-IS-08, NF-IS-09, NF-PO-04

fsni-sink-gs

Rol:	Servidor para Graph Streaming (Gephi)
Dependencias:	
Descripción:	1. Sobrescribe algunos métodos de la clase libfsni2graph para recibir notificación de creación/modificación/destrucción de un nodo o arista 2. Cada evento genera una cadena JSON que es escrita en un socket (ver documentación de Graph Streaming de Gephi para ver qué es válido aquí).
Datos:	▪ [in] libfsni stream ▪ [out] JSON/HTTP compatible Gephi
Recursos:	▪ libstdc++
Origen:	F-SR-32, F-SR-34, F-SR-35, NF-IC-02, NF-IC-08, NF-IS-08, NF-IS-09, NF-PO-04

3.5.2. Componentes de terceros

graphviz

Rol:	Generar una imagen de un grafo DOT
Dependencias:	
Descripción:	Uno de sus *layout commands* es ejecutado bajo demanda por el usuario para generar una imagen.
Datos:	▪ [in] entrada DOT ▪ [out] imagen generada
Recursos:	▪ imlib/cairo
Origen:	F-SR-33

Gephi

Rol:	Abrir un fichero DOT o ver un grafo en tiempo real
Dependencias:	
Descripción:	Ejecutado bajo demanda para abrir un DOT o ver un grafo en tiempo real.
Datos:	▪ [in] entrada DOT o JSON/HTTP
Recursos:	▪ JDK>=1.7
Origen:	F-SR-34, NF-IC-08

3.6. Matrices de trazabilidad

Esta sección incluye las matrices de trazabilidad "requisitos usuario–requisitos software" y "requisitos software–componentes". Éstas aseguran que no hay ningún requisito no representado en el diseño.

3.6.1. Trazabilidad SR-UR

Las matrices de trazabilidad "requisitos usuario–requisitos software" se incluyen en las tablas 3.1, 3.2, 3.3 y 3.4.

	CA-UR-01	CA-UR-02	CA-UR-03	CA-UR-04	CA-UR-05	CA-UR-06	CA-UR-07	CA-UR-08	CA-UR-09	CA-UR-10	CA-UR-11	CA-UR-12	CA-UR-13	CA-UR-14	CA-UR-15	CA-UR-16	CA-UR-17	CA-UR-18	CA-UR-19	CA-UR-20
F-SR-01	•								•											
F-SR-02		•						•												
F-SR-03		•																		
F-SR-04		•							•											
F-SR-05		•						•												
F-SR-06		•																		
F-SR-07			•																	
F-SR-08			•																	
F-SR-09		•																		
F-SR-10				•	•	•	•													
F-SR-11				•																
F-SR-12				•																
F-SR-13				•		•	•													
F-SR-14					•															
F-SR-15								•												
F-SR-16		•						•		•										
F-SR-17											•									
F-SR-18											•									
F-SR-19												•								
F-SR-20													•							
F-SR-21														•						
F-SR-22														•						
F-SR-23														•						
F-SR-24														•						

Tabla 3.1: Matriz de trazabilidad F-SR/CA-UR (1 de 2)

	CA-UR-01	CA-UR-02	CA-UR-03	CA-UR-04	CA-UR-05	CA-UR-06	CA-UR-07	CA-UR-08	CA-UR-09	CA-UR-10	CA-UR-11	CA-UR-12	CA-UR-13	CA-UR-14	CA-UR-15	CA-UR-16	CA-UR-17	CA-UR-18	CA-UR-19	CA-UR-20
F-SR-25													•							
F-SR-26													•							
F-SR-27														•	•					
F-SR-28														•						
F-SR-29																•				
F-SR-30		•																		•
F-SR-31		•																		•
F-SR-32																	•	•		
F-SR-33																	•			
F-SR-34																		•		
F-SR-35																		•		

Tabla 3.2: Matriz de trazabilidad F-SR/CA-UR (2 de 2)

	RE-UR-01	RE-UR-02	RE-UR-03	RE-UR-04	RE-UR-05	RE-UR-06	RE-UR-07	RE-UR-08	RE-UR-09	RE-UR-10	RE-UR-11	RE-UR-12	RE-UR-13	RE-UR-14	RE-UR-15	RE-UR-16	RE-UR-17
NF-IC-01	•																
NF-IC-02		•															
NF-IC-03		•															
NF-IC-04			•														
NF-IC-05			•														
NF-IC-06				•													
NF-IC-07				•													
NF-IC-08					•												
NF-IH-01						•									•		
NF-IH-02						•											
NF-IH-03						•											
NF-IS-01							•										
NF-IS-02							•										
NF-IS-03							•										
NF-IS-04							•										
NF-IS-05							•										

Tabla 3.3: Matriz de trazabilidad NF-/RE-UR (1 de 2)

	RE-UR-01	RE-UR-02	RE-UR-03	RE-UR-04	RE-UR-05	RE-UR-06	RE-UR-07	RE-UR-08	RE-UR-09	RE-UR-10	RE-UR-11	RE-UR-12	RE-UR-13	RE-UR-14	RE-UR-15	RE-UR-16	RE-UR-17
NF-IS-06						•											
NF-IS-07						•											
NF-IS-08							•										
NF-IS-09								•									
NF-IU-01									•								
NF-IU-02									•								
NF-PO-01										•		•					
NF-PO-02										•		•					
NF-PO-03										•		•					
NF-PO-04											•						
NF-PO-05										•							
NF-SE-01													•				
NF-SE-02													•				
NF-SE-03													•				
NF-TM-01															•		
NF-TM-02														•	•		
NF-TM-03														•	•		
NF-TM-04																•	

Tabla 3.4: Matriz de trazabilidad NF-/RE-UR (2 de 2)

3.6.2. Trazabilidad SR-Componentes

Las matrices de trazabiliad “requisitos software–componentes” se incluyen en las tablas 3.5, 3.6 y 3.7.

	F-SR-01	F-SR-02	F-SR-03	F-SR-04	F-SR-05	F-SR-06	F-SR-07	F-SR-08	F-SR-09	F-SR-10	F-SR-11	F-SR-12	F-SR-13	F-SR-14	F-SR-15	F-SR-16	F-SR-17	F-SR-18	F-SR-19	F-SR-20	F-SR-21	F-SR-22	F-SR-23	F-SR-24	F-SR-25	F-SR-26	F-SR-27	F-SR-28	F-SR-29	F-SR-30
Gephi																														
filters						•	•																							
fsni-sink-dump																		•												
fsni-sink-graph																														
fsni-sink-gs																														
fsni-sink-gui																			•	•	•	•	•	•						
fsni-sink-streams																										•	•	•		
fsni-src-fuse	•																													
fsni-src-kctl			•																											
fsni-src-kiocalsa																														
fsni-src-klinux			•	•	•			•						•	•	•	•											•		
fsni-src-pt	•	•																												
graphviz																														
libfsni									•	•	•	•	•																	
libfsni2graph																														

Tabla 3.5: Matriz de trazabilidad SR-/componentes (1 de 3)

	F-SR-31	F-SR-32	F-SR-33	F-SR-34	F-SR-35	NF-IC-01	NF-IC-02	NF-IC-03	NF-IC-04	NF-IC-05	NF-IC-06	NF-IC-07	NF-IC-08	NF-IH-01	NF-IH-02	NF-IH-03	NF-IS-01	NF-IS-02	NF-IS-03	NF-IS-04	NF-IS-05	NF-IS-06	NF-IS-07	NF-IS-08	NF-IS-09	NF-IU-01	NF-IU-02	NF-PO-01	NF-PO-02	NF-PO-03
Gephi			•									•																		
filters																							•							
fsni-sink-dump						•																	•	•	•					
fsni-sink-graph		•	•			•																	•	•						
fsni-sink-gs	•		•	•		•						•											•	•						
fsni-sink-gui						•																	•			•				
fsni-sink-streams						•																	•	•						
fsni-src-fuse					•	•										•		•					•	•	•		•			
fsni-src-kctl					•														•	•	•		•	•						
fsni-src-kiocalsa	•					•															•		•							
fsni-src-klinux						•										•		•	•	•	•		•						•	
fsni-src-pt					•	•																	•	•	•			•		
graphviz		•																												
libfsni							•	•	•	•	•				•							•	•							
libfsni2graph	•												•	•	•							•								

Tabla 3.6: Matriz de trazabilidad SR-/componentes (2 de 3)

	NF-PO-04	NF-PO-05	NF-SE-01	NF-SE-02	NF-SE-03	NF-TM-01	NF-TM-02	NF-TM-03	NF-TM-04
Gephi									
filters									
fsni-sink-dump	•								
fsni-sink-graph	•								
fsni-sink-gs	•								
fsni-sink-gui	•								
fsni-sink-streams	•								
fsni-src-fuse		•			•	•			
fsni-src-kctl									
fsni-src-kiocalsa		•							
fsni-src-klinux		•	•				•		
fsni-src-pt				•					
graphviz									
libfsni									•
libfsni2graph	•								

Tabla 3.7: Matriz de trazabilidad SR-/componentes (3 de 3)

Capítulo 4

Implementación

En este capítulo se incluyen detalles de implementación de algunos componentes del proyecto (los sencillos se han omitido).

Puede encontrar código importante en el capítulo D.

4.1. libfsni

Esta sección incluye una vista general de la implementación de libfsni. Ésta provee interfaces para la serialización/deserialización de mensajes. En la figura 4.1 puede apreciarse el rol de libfsni en el proyecto.

Como puede verse en la figura 4.2, el productor/consumidor hará uso de funciones exportadas por libfsni.

4.1.1. Cabeceras de mensaje

En el siguiente listado[1] se muestra la definición del tipo `struct msghdr` que representa un mensaje. Para más información sobre el formato en el que lib representa los mensajes vea la sección A en la página 93.

[1] Sólo están los miembros que el usuario debería acceder directamente. Para acceder a origen y la parte dependiente de mensaje use las macros fsni_get_msghdr_src() y fsni_get_msg_any() documentadas más abajo. La definición completa puede verse en libfsni/libfsni.h

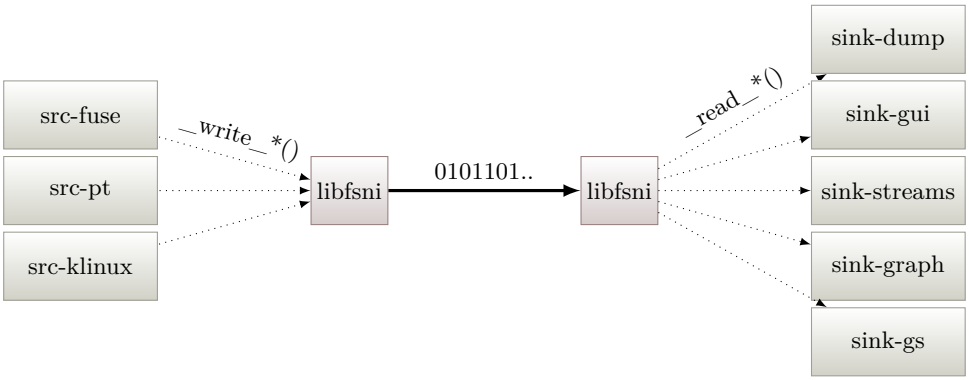


Figura 4.1: Rol de libfsni en el proyecto

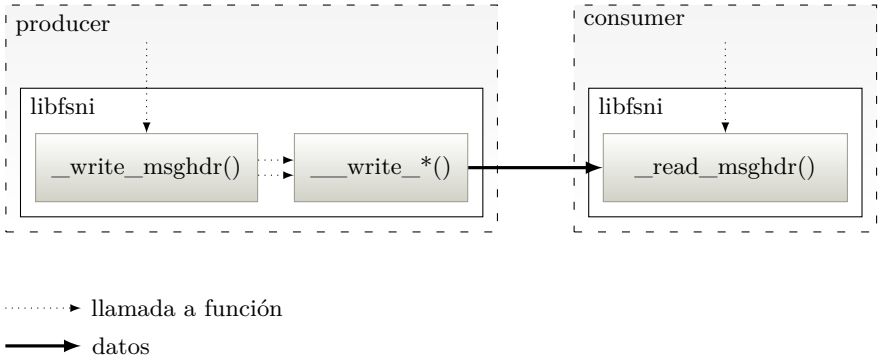


Figura 4.2: Llamadas a libfsni

```
1  struct msghdr {
2          __u8  type;
3          __u8  flags;
4          __u16 err;
5          __u32 seq;
6          __u32 length;
7          /* ... */
8  } __attribute__((packed));
```

Los miembros de `struct msghdr` se describen a continuación:

type Tipo del mensaje; determina qué datos dependientes del tipo están disponibles. La parte dependiente de tipo puede llevar datos adicionales e.g. en el caso de una lectura: fichero, desplazamiento, bytes transferidos... La tabla 4.2 lista los valores válidos. Para más información lea la sección A.5 que incluye una referencia de cada tipo.

flags Indicadores asociados a este mensaje. Los indicadores disponibles y su significado se describen en la tabla 4.1. Por favor, note que:

- Los tipos que se documentan con el flag `G` llevan éste siempre activo (anidados).
- libfsni puede truncar el payload de cualquier mensaje y activar el flag `T` en consecuencia.

err *errno* en operaciones que terminen con error.

seq Número de secuencia; permite detectar mensajes perdidos. Esto es un espacio circular de 32-bit que recuerda a algunos protocolos de enrutamiento[JD06], vea la figura 4.3.

length Tamaño en bytes, payload incluido.

4.1.2. Orden de bytes

Es seguro usar libfsni para generar un log en una arquitectura diferente a la que posteriormente lo leerá. libfsni escribe los mensajes en el orden de bytes nativo de la arquitectura, sin embargo el consumidor detecta si el orden de bytes de la arquitectura que generó el log es diferente e intercambia automáticamente éstos.

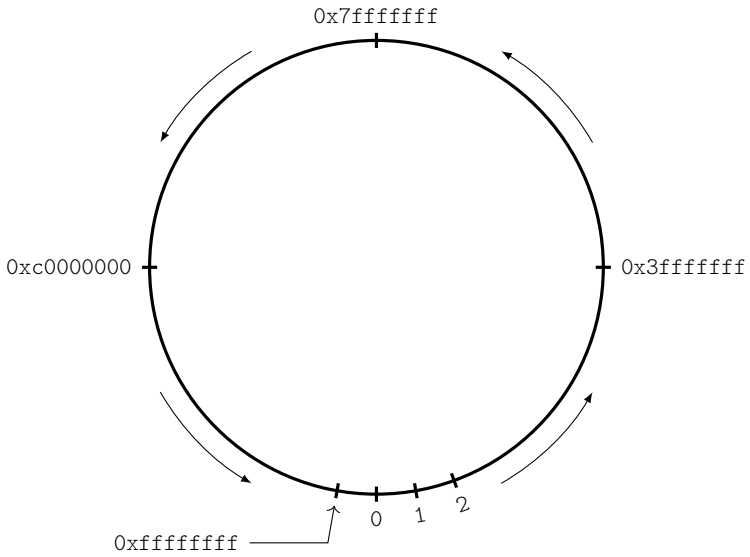


Figura 4.3: Espacio circular de 32-bit de seq

Flag	C #define	Descripción
A	F_AIO	IO asíncrona
S	F_SPLICE	El proceso hizo una llamada a splice()
W	F_WRITE	Señaliza una escritura
T	F_TRUNCATED	Payload truncado por lib
G	F_GROUP	Si G=1 este mensajes es extensión de uno anterior que tiene G=0

Tabla 4.1: Flags disponibles para los mensajes

Tipo	Flags	Payload
OPEN [0]	`---T-`	Argumento ⟨pathname⟩ pasado a open()
RELEASE [1]	`-----`	
RW [2]	`-SWT-`	Buffer transferido
RW_ITER [3]	`ASW--`	
IOVEC [4]	`---TG`	Buffer transferido
IOCTL [5]	`-----`	
IOCTL_DATA [6]	`---TG`	Dependiente del argumento *cmd* de ioctl()
MMAP [7]	`-----`	
MUNMAP [8]	`-----`	
VM_FAULT [9]	`--WT-`	La página que se accedía cuando ocurrió el fallo
WRITEPAGE [10]	`--WT-`	La página que se volcó a disco

Tabla 4.2: Tipos de mensaje de libfsni

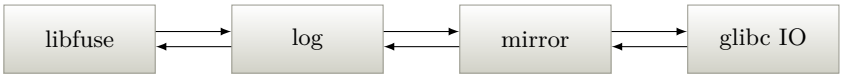


Figura 4.4: Orden de llamadas en fsni-src-fuse

4.1.3. Seguridad en hilos

`fsni_write_msghdr()` no escribe mensajes de forma atómica —hace 2 llamadas a `__write_xxx()` si el mensaje tiene payload. Por tanto, debe sincronizar las escrituras manualmente con un mutex/spinlock.

Si va a escribir un grupo de mensajes debe protejer todo el bloque de llamadas `fsni_write_msghdr()` para asegurar que se escriben atómicamente.

```
1 /* adquirir lock */
2 fsni_write_msghdr() /* n llamadas */
3 /* liberar lock */
```

En el caso del consumidor, si está compilando libfsni en un sistema GNU, ésta hará uso automáticamente de `flockfile()`, `funlockfile()` y `fread_unlocked()`, por lo que no es necesaria sincronización por parte del usuario.

4.2. filters

Éste implementa un filtro de captura basado en path que será usado por todos los SRC. Usa `include/linux/list.h` del kernel Linux portado a espacio usuario.

Debido a que es sencillo, no se usará más espacio aquí. Implementado en `include/filters.h` (funciones inline).

4.3. fsni-src-fuse

fsni-src-fuse es la implementación más simple de SRC: un sistema de ficheros en espacio de usuario usando FUSE que replica el directorio / a la vez que loguea las operaciones usando libfsni. En la figura 4.4 puede apreciarse el orden de las llamadas.

A continuación se describe el camino de una ejecución para `cat /mnt/fuse/etc/resolv.conf` (ver figura 4.5):

1. cat llama a glibc `open()`, que invoca la llamada a sistema manejada en kernel por `sys_open()`.
   VFS invoca la operación `open()` (de file_operations que provee el módulo de kernel FUSE). La implementación de fuse es serializar y pasar a Userland.

2. El proceso fsni-src-fuse deserializa e invoca a `open()` (de fuse_file_operations), que está implementado por `log_open()`.

3. `log_open()` invoca a la capa inferior `mirror_open()`.

4. `mirror_open()` invoca a `open()` pasando como argumento `/etc/resolv.conf`.

5. `open()` es servido por el sistema de ficheros destino.

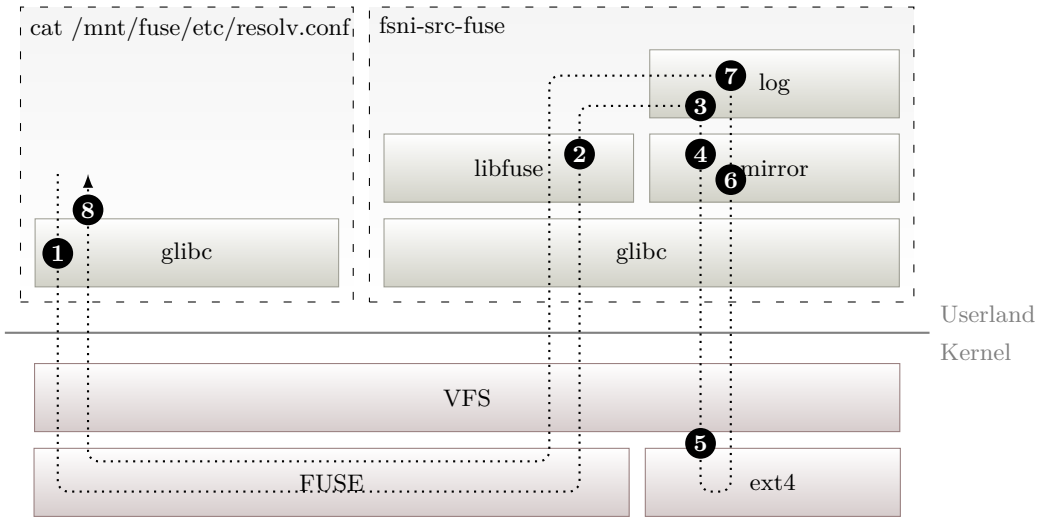


Figura 4.5: Camino de una ejecución de fsni-src-fuse

6. `mirror_open()` devuelve el resultado.

7. `log_open()` loguea la operación; cuando retorna, libfuse serializa el resultado.

8. `sys_open()` retorna con el resultado anterior.

4.3.1. Capa mirror

Ésta implementa la mayoría de las operaciones `fuse_operations` excepto `ioctl()`, por cuestiones de seguridad (ver la sección 4.3.3 para más información).

El código no depende de log, por lo que `src/src-fuse/main.c` podría ser ajustado para espejar / sin loguear las operaciones.

Está basado en `fusexmp_fh.c` (Miklos Szeredi, Sebastian Pipping) del proyecto FUSE (`http://fuse.sourceforge.net/`), con algunas modificaciones.

Como ejemplo se incluye la implementación de `mirror_mkdir` en el listado 4.1. La implementación del resto de operaciones `fuse_operations`, con alguna excepción, no es muy diferente de lo que puede verse en el listado anterior.

Listado 4.1: fusemirror_mkdir()

```
1 static int mirror_mkdir(const char *path, mode_t mode)
2 {
3         return (mkdir(path, mode) == -1) ? -errno : 0;
4 }
```

4.3.2. Capa log

La capa log se interpone entre algunas de las `fuse_operations` y loguea las operaciones usando libfsni. El listado 4.2 contiene la función que reemplaza las operaciones FUSE.

Listado 4.2: src/src-fuse/log.c:log_hook_fuse_operations()

```
1  struct fuse_operations *log_hook_fuse_operations(struct log_fs *fs,
2                                                   struct fuse_operations *orig_ops) {
3          struct fuse_operations *ops = malloc(sizeof(struct fuse_operations));
4
5          if (!ops)
6                  return NULL;
7
8          /* as most operations remain unchanged: copy operations, then fixup */
9          memcpy(ops, orig_ops, sizeof(struct fuse_operations));
10         ops->open = log_open;
11         ops->read = log_read;
12         ops->write = log_write;
13         ops->release = log_release;
14         ops->init = log_init;
15         ops->destroy = log_destroy;
16         ops->create = log_create;
17
18         /* set to NULL if user requested buffer copying */
19         ops->read_buf = (fs->flags & LOGF_BUFFERS)  ? NULL : log_read_buf;
20         ops->write_buf = (fs->flags & LOGF_BUFFERS) ? NULL : log_write_buf;
21
22         fs->orig_ops = orig_ops;
23         return ops;
24 }
```

4.3.3. Cuestiones de seguridad

Los argumentos de `ioctl()` pueden incluir direcciones de memoria que serán origen/destino de copias de memoria desde espacio kernel. Debido a que esto es dependiente del número de ioctl, es imposible determinar si se terminará copiando regiones arbitrarias de memoria del proceso fsni-src-fuse.

ioctl()... ¿Es posible?

FUSE implementa un mecanismo para la copia de datos durante una llamada `ioctl()`, pero es necesario saber qué memoria copiar, lo que depende del número de IOCTL. Para más información vea la documentación del proyecto FUSE en `http://fuse.sourceforge.net/`

Esto permitiría soportar ioctls cargando objetos compartidos ELF (.so).

4.4. fsni-src-pt

fsni-src-pt usa `ptrace(PTRACE_SYSCALL)` para trazar las llamadas a sistema ejecutadas por otro proceso (ver tabla 4.3 para saber que syscalls se manejan), de modo similar a la utilidad `strace` y por tanto con la misma penalización de rendimiento. Ha sido implementado en C++ para usar la STL (C++11 debido a que necesita `std::shared_ptr`).

La clase pt_syscall maneja la traza de llamadas a sistema con `ptrace()`. Se soporta la traza de procesos multithread haciendo que `ptrace()` se adjunte automáticamente a nuevos threads con la opción `PTRACE_O_TRACECLONE`.

En la figura 4.6 puede verse el camino de la ejecución de fsni-src-pt:

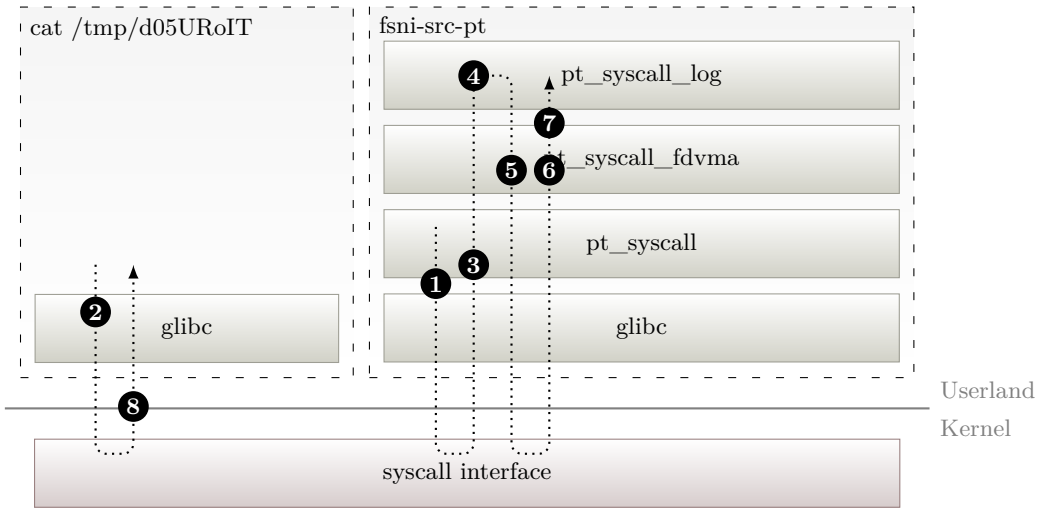


Figura 4.6: Camino de una ejecución de fsni-src-pt

1. Se solicita la parada del proceso trazado en la siguiente salida de una llamada a sistema y se queda esperando
2. El proceso invoca la syscall `open()`
3. Se invoca la función miembro virtual `pt_syscall_fdvma::on_syscall_exit()`
4. Ésta invoca a la función miembro virtual `pt_syscall_log::sys_open_creat()`...
5. `pt_syscall_fdvma::sys_open_creat()` crea un objeto pt_file para el descriptor de fichero devuelto (si éxito)
6. Varias llamadas `ptrace(PTRACE_PEEKDATA)` pueden ocurrir para copiar la cadena pasada como primer argumento.
7. `pt_syscall_log::sys_open_creat()` loguea la operación.
8. La llamada a sistema retorna.

4.4.1. Manteniendo el estado

Algunas llamadas a sistema devuelven un valor que será pasado a otras llamadas a sistema. Tal es el caso de las siguientes[2]:

- Requieren un descriptor devuelto por open()/creat() [struct file]: close, lseek, _llseek, dup, dup2, dup3, fcntl, fcntl64, mmap, mmap2, read, write, pread64, pwrite64, readv, writev, preadv, pwritev, ioctl
- Requieren una dirección devuelta por mmap() [struct vm_area_struct]: munmap

En este caso parte del estado de estos objetos tendrá que ser duplicado en espacio de usuario (clase pt_syscall_fdvma).

[2]La estructura que mantiene el estado en kernel semuestra entre [ ]

Syscall	Handler
open, creat	`pt_syscall_log::sys_open_creat()`
close	`pt_syscall_log::sys_close()`
lseek	`pt_syscall_fdvma::sys_lseek()`
_llseek	`pt_syscall_fdvma::sys__llseek()`
dup, dup2, dup3	`pt_syscall_fdvma::sys_dup_dup2_dup3()`
fcntl, fcntl64	`pt_syscall_fdvma::sys_fcntl_fcntl64()`
mmap, mmap2	`pt_syscall_log::sys_mmap_mmap2()`
munmap	`pt_syscall_log::sys_munmap()`
read, write, pread64, pwrite64	`pt_syscall_log::sys_read_write_pread64_pwrite64()`
vreadv, writev, preadv, pwritev	`pt_syscall_log::sys_readv_writev_preadv_pwritev()`
io_submit	`pt_syscall_log::sys_io_submit()`
ioctl	`pt_syscall_log::sys_ioctl()`

Tabla 4.3: Llamadas a sistema interceptadas por ptrace

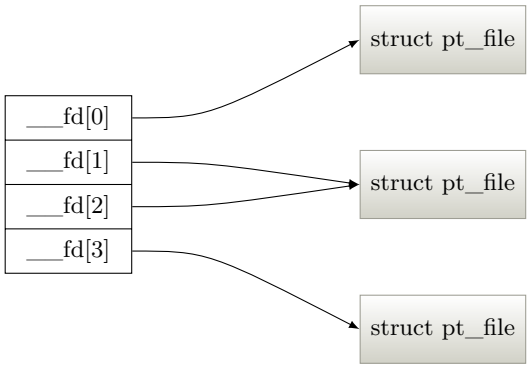


Figura 4.7: Tabla de descriptores de fichero

Descriptores de fichero

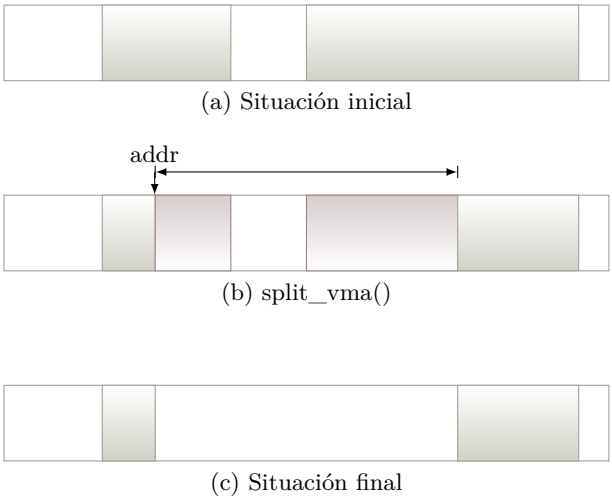
fsni-src-pt mantiene una tabla de descriptores de fichero con punteros a `struct pt_file` de tipo `std::vector<std::shared_ptr<pt_file> >` (ver figura 4.7):

```
struct pt_file {
        /* arguments passed to open(2) */
        std::string pathname;
        int         flags;
        mode_t      mode;

        loff_t offset;  /* file offset; system call handlers should update this */

        void *private_data;

        pt_file(std::string &_p, int _f, int _m)
          : pathname(_p), flags(_f), mode(_m), offset(0) { };
};
```



(a) Situación inicial

(b) split_vma()

(c) Situación final

Figura 4.8: pt_syscall_fdvma::munmap()

VMA

El caso de una VMA es más complicado de manejar, ya que es posible invocar a munmap() para *desmapear* un intervalo de éstas, que podría partir VMA existentes. Para esto, se portó la implementación del kernel Linux (`mm/mmap.c`) a la STL.

Un `std::map<unsigned long, pt_vma *>` mantendrá los mapeos actuales. La figura 4.8) es una posible llamada a `pt_syscall_fdvma::munmap()`.

4.4.2. Alternativa: LD_PRELOAD

Se ha usado `ptrace()` para interceptar llamadas a sistema de otro proceso/copiar memoria desde su espacio de direcciones, pero hay una alternativa: si usa glibc para hacer una llamada a sistema, estará invocando a una función de biblioteca que hará ésta en su nombre:

```
[...]
fd = open("pathname", O_RDONLY);
    [glibc open]
    movl $5, %eax          /* número syscall open() */
    movl "pathname", %ebx  /* dirección de "pathname" */
    movl 0, %ecx           /* O_RDONLY */
    sysenter               /* VDSO: int 0x80 si sysenter no soportado
                              retorno en %eax */
[...]
```

ld-linux.so usa la variable de entorno `LD_PRELOAD` (ver ld.so(8)) para cargar objetos compartidos ELF adicionales que pueden usarse para "pisar" símbolos existentes (shims).

Es posible escribir un reemplazo de `open()` e incluirlo en *alt-open.so*. Si ejecutamos un proceso enlazado dinamicamente con `LD_PRELOAD=alt-open.so`:

- ld-linux.so cargará nuestra biblioteca en el espacio de direcciones del proceso.
- Parcheará las llamadas a `open()` con la dirección de alt-open.so:open().

- Para no romper el comportamiento, nuestra implementación debería encargarse de llamar a glibc `open()`.

De este modo, cualquier llamada a sistema vía glibc puede ser capturada. Esta alternativa no tiene las penalizaciones de rendimiento asociadas a ptrace(), pero esta restringida a los siguientes usos:

- Sólo es posible interceptar llamadas a sistema invocadas usando funciones de biblioteca (glibc).
- Aún si éste fuera el caso, está limitado a ejecutables compilados dinámicamente contra glibc.

4.5. fsni-src-klinux

Éste atiende solicitudes desde userspace para trazar un proceso (TGID) o grupo de ellos (COMMand name) y exporta un *libfsni stream* via un dispositivo de caracteres. Cada solicitud hecha podrá ser suscriptor de los eventos de n publicadores —un publicador aquí es un puntero a cualquier objeto del kernel (file, vm_area_struct, page...).

4.5.1. Vista general

Muchas estructuras del kernel contienen un puntero a una tabla de operaciones ($\simeq$ vtable para C++); el objetivo será parchear éstas en nombre de un publicador. Además, cuando todos los publicadores hayan terminado se restaurará la tabla de operaciones original —veremos que ésto es exactamente lo que hace la capa koh_mgmt.

La idea es parchear las llamadas a sistema que crean un objeto `struct file` —open(), openat(), creat()— y antes de retornar a espacio usuario:

- Obtener la lista de suscriptores para este fichero
- Si hay suscriptores, parchear la tabla de operaciones para este objeto file[3].

4.5.2. Parcheo de llamadas a sistema

Parchear una llamada a sistema es cuestión de cambiar `sys_call_table[__NR_xxx]` para hacerlo apuntar a nuestra implementación pero...

El kernel no exporta el símbolo sys_call_table aunque el manejador de llamada a sistema sabe su dirección, ya que usa esa tabla de saltos para invocar a "sys_xxx".

Aquí la idea[sd01] es inspeccionar la memoria desde la dirección del manejador para encontrar $0xff0x140x85$ —opcode de `call *sys_call_table(,%eax,4)`. La dirección debería seguir al byte SIB (0x85).

Éste está en la sección .rodata (sólo lectura) aunque como estamos en ring-0, modificamos su entrada en la tabla de páginas.

[3]Normalmente, las tablas de operaciones son constantes y compartidas por muchos objetos del mismo tipo, por lo que se debería cachear la versión parcheada (tcache).

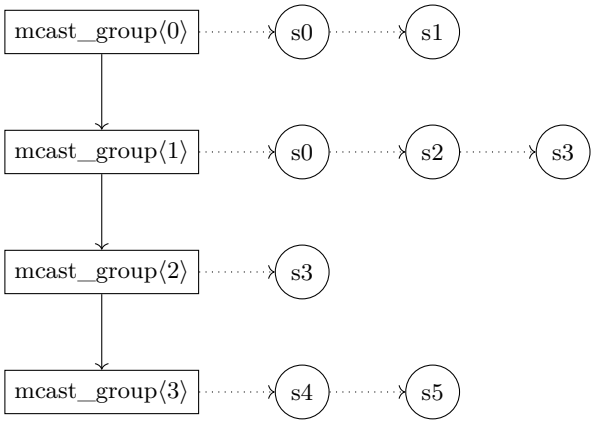


Figura 4.9: Lista de grupos multicast activos (___mcast_list)

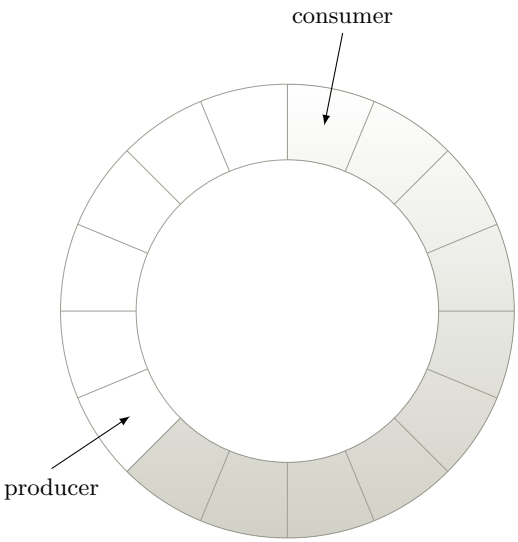


Figura 4.10: Buffer circular usado en fsni-src-klinux

4.5.3. Diseño del módulo

Esta sección está dedicada a describir todas las capas del módulo que se ha escrito:

tcache Cacheo de tablas de operaciones parcheadas.

mcast Gestión de grupos multicast (de suscriptores); ver figura 4.9

koh_mgmt Gestión de Hook de Objetos de Kernel[Kon07] y mapeo de publicador → grupo

chrdev Interfaz con userspace (dispositivo de carácter); ver figura 4.10

sc_hook Soporte de hooks de llamada a sistema (sd & devik)

hooks Implementación de "xxx_operations" y reemplazos de syscalls

En la figura 4.11 puede verse el funcionamiento del módulo (en los puntos 10 y 11 se escribe o lee respectivamente el buffer circular). El camino de ejecución corresponde a las llamadas open() seguida de read() en el proceso trazado:

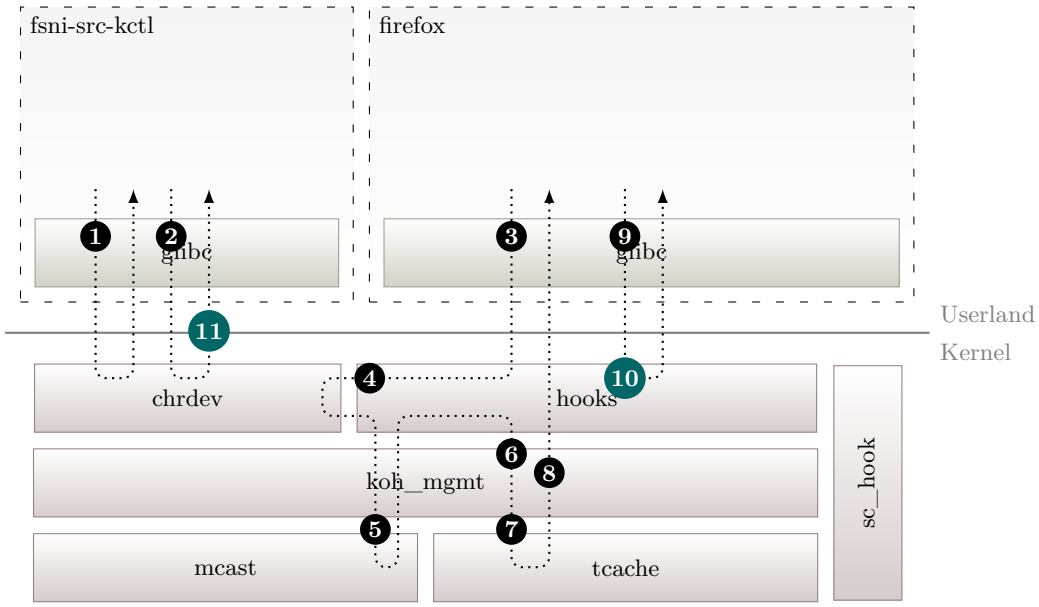


Figura 4.11: Camino de ejecución para fsni-src-klinux

1. fsni-src-kctl abre el dispositivo `/dev/fsni-src-klinux` y hace una llamada ioctl() para establecer los parámetros de la traza y activarla.
2. fsni-src-kctl se queda bloqueado esperando la llegada de datos desde el módulo de kernel
3. El proceso trazado (firefox) hace una llamada sistema `open()`...
4. ...que es atendida por nuestro manejador de llamada a sistema `____sys_open`. Éste invoca a la implementación original de `sys_open`. Cuando `sys_open` retorna se mira si hay suscriptores que deban recibir eventos para este fichero.
5. Se crea un grupo multicast que incluya a los suscriptores anteriores.
6. Si hay suscriptores para este fichero, se invoca a `koh_hook` para parchear la tabla de operaciones `file_operations` de este objeto file.
7. Se consulta la cache de tablas parcheadas; si esta tabla no se ha parcheado antes, se hace ahora.
8. Se cambia el puntero `f_op` del objeto file para apuntar a nuestra tabla y se añaden entradas en `__koh_rb_object` (mapa de objetos hookeados) y `__koh_rb_delivery` (mapa de entrega).
9. El proceso invoca la llamada a sistema `read()`.
10. Nuestra operación `read()` de `file_operations` es invocada: se llama a la implementación original y se escribe un mensaje en el buffer circular de cada suscriptor. Se despiertan los suscriptores.
11. Se leen datos disponibles desde el buffer circular.

En el apéndice D puede verse el código competo de fsni-src-klinux.

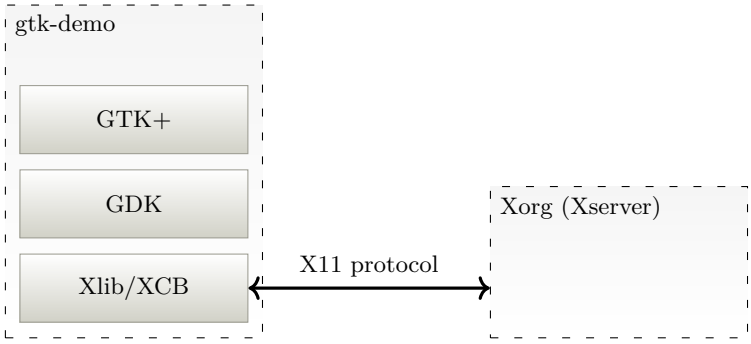


Figura 4.12: GTK+/X11

4.6. fsni-sink-gui

GTK+ es un toolkit para el desarrollo de interfaces gráficas de usuario sobre X11. GDK es la interfaz entre gráficos de alto nivel y gráficos de bajo nivel (Xlib), de manera que portando GDK a una determinada plataforma permite a GTK+ correr sobre ella. Posteriormente se han escrito implementaciones de GDK para Win32, Mac OS X y otros. Para más información vea `http://www.gtk.org/`.

La figura 4.12 muestra el papel de cada componente.

fsni-sink-gui hace uso de GtkBuilder, que permite construir la interfaz a partir de un fichero XML, habitualmente generado con una herramienta como Glade.

4.6.1. Captura desde fsni-sink-gui

fsni-sink-gui debe descartar los payloads de los mensajes y tenerlos disponibles más adelante bajo demanda del usuario.

Para esto, fsni-sink-gui crea un fichero temporal y dos threads (*external_proc_stdout_thread* y *external_proc_tee_thread*) al iniciar la captura; cuando el usuario solicita el payload de un mensaje, éste lee el payload desde el fichero temporal usando `pread()`. En la figura 4.13 se puede ver esta situación.

external_proc_stdout_thread se encarga de:

- Copiar (sin consumir) bytes `__user_data->child_stdout` → `__user_data->pipefd[1]` (tee).
- Mover bytes desde `__user_data->child_stdout` → fichero temporal (splice).

external_proc_tee_thread por su parte se encarga de:

- Usar libfsni para leer mensajes desde el otro extremo de la tubería `__user_data->pipefd`.
- Encolar en un GAsyncQueue el mensaje para añadirlo a GtkListStore. La recogida se hace con `gdk_threads_add_timeout`.

4.7. fsni-sink-gs

En la figura 4.14 se puede ver cuando Gephi está conectado a fsni-sink-gs.

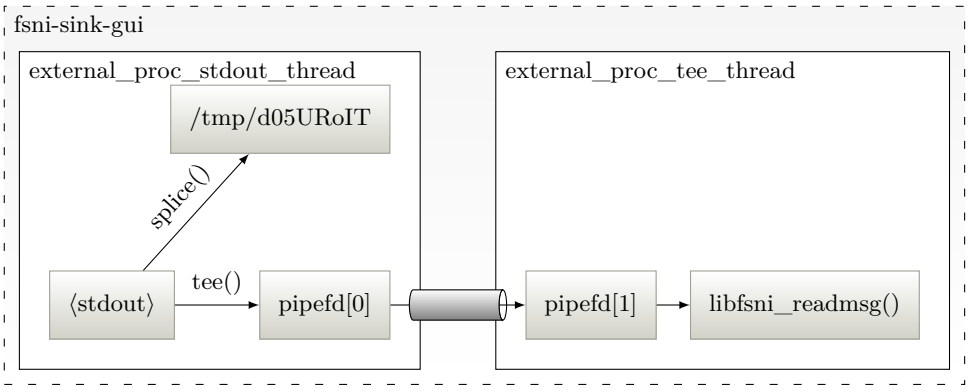


Figura 4.13: fsni-sink-gui durante una captura

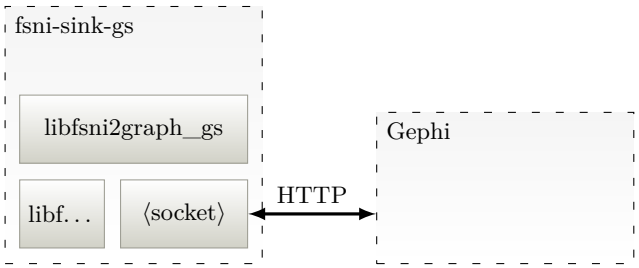


Figura 4.14: Conectando a Gephi

Capítulo 5

Pruebas y evaluación

En este capítulo se incluyen las pruebas realizadas para asegurar el funcionamiento del software y la evaluación de rendimiento de los métodos de captura incluidos.

5.1. Pruebas de verificación

Esta sección contiene información de las pruebas realizadas como parte del desarrollo para verificar que el comportamiento es conforme a la especificación de requisitos del software.

5.1.1. Entorno de prueba

En esta sección se describe el entorno software/hardware sobre el que se ejecutaron las pruebas durante el desarrollo.

El software queda descrito en la lista siguiente:

- GNU/Linux (kernel 4.0.9) x86_64 —archlinux
- Ningún framework de seguridad está habilitado (SELinux, apparmor...)
- glibc 2.13
- GNU autoconf 2.69, automake 1.11.6
- gcc 4.6.3
- FUSE 2.9
- GTK+ 3.16
- Gephi 0.8.2

Las pruebas de rendimiento de la sección 5.2 se ejecutaron sobre el hardware siguiente:

- Procesador Intel Core 2 1.86GHz
- Chipset Intel 945GM
- 1GB RAM DDR2-667
- Disco duro Western Digital Scorpio Black 750GB

5.1.2. Criterio de aceptación

Esta sección especifica el criterio de aceptación de las pruebas descritas en la sección 5.1.3. Debido al alcance del proyecto, las pruebas de dicha sección se limitan a verificar funcionalidad genérica que debe estar disponible en cualquier caso.

Para dar por válida una prueba, debe ser cierto que:

- La prueba se ejecuta en un entorno software equivalente al descrito en la sección 5.1.1.
- La prueba se ha ejecutado siguiendo el procedimiento que se ha descrito, especificado en *descripción* y *entrada*.
- El resultado de la prueba coincide con el esperado, especificado en *salida*.
- No hay restricción del orden de ejecución de las pruebas, excepto en aquellas que así lo especifiquen. En este caso, deberá ejecutar primero la prueba referenciada, ya que alguna de sus salidas es usada por la prueba a ejecutar.

5.1.3. Especificación de casos de prueba

En esta sección se describen los casos de prueba usados para asegurar el funcionamiento de los componentes del proyecto.

Debido a que el log generado es dependiente del objetivo de la captura, es necesario especificar un programa que genere una secuencia de mensajes predecible. El código del programa `test.c` usado en algunas pruebas es esencialmente equivalente a[1]:

```
/* #include <..> */
#define BUFSIZE 4096

int main(int argc, char *argv[])
{
        int fd1, fd2;
        long __pagesize = sysconf(_SC_PAGE_SIZE);
        char buf[BUFSIZE], *mmap_addr;
        io_context_t ctx;
        struct io_event events[1];
        int inode_flags;

        fd1 = open("/dev/urandom", O_RDONLY);
        fd2 = open("/tmp/.test.out", O_CREAT | O_TRUNC, 0666);

        /* E/S */
        read(fd1, buf, BUFSIZE);
        write(fd2, buf, BUFSIZE);
        close(fd1);

        /* E/S vectorizada */
        struct iovec iov[] = { { .iov_base = buf, .iov_len = BUFSIZE},
                               { .iov_base = buf, .iov_len = BUFSIZE} };
        preadv(fd2, iov, 2, 0);
        pwritev(fd2, iov, 2, BUFSIZE);

        /* E/S asíncrona */
        struct iocb iocb = { .aio_fildes = fd2, .aio_lio_opcode = IOCB_CMD_PREAD,
                             .aio_buf = buf, .aio_offset = 0, .aio_nbytes = BUFSIZE };
        struct iocb *ios[] = { &iocb };
        io_setup(16, &ctx);
```

[1]Para el código completo vea support/test.c

```
32          io_submit(ctx, 1, ios);
33          io_getevents(ctx, 1, 1, events, NULL);
34          io_destroy(ctx);
35
36          /* ioctl */
37          ioctl(fd2, FS_IOC_GETFLAGS, &inode_flags);
38
39          /* mmap */
40          mmap_addr = mmap(NULL, __pagesize, PROT_READ | PROT_WRITE,
41                  MAP_SHARED, fd2, 0);
42          memcpy(buf, mmap_addr, BUFSIZE);
43          memcpy(mmap_addr, buf, BUFSIZE);
44          munmap(mmap_addr, __pagesize);
45          close(fd2);
46
47          return 0;
48  }
```

Los casos de prueba se definen usando la plantilla de la figura 5.1 que especifica:

Identificador será CP-XX donde XX es un número de secuencia iniciado en 01. Encabeza la plantilla.

Descripción del caso de uso, especificando qué es lo que se prueba.

Entrada entradas provistas por el usuario.

Salida refiere la salida esperada de la prueba.

Entorno requisitos del entorno en el que se ejecutará la prueba.

CP-XX

Descripción:

Entrada:

Salida:

Entorno: n/a

Figura 5.1: Plantilla de casos de prueba

Pruebas no documentadas

En la fase de desarrollo se han hecho otras pruebas no documentadas aquí y que estaban dirigidas a arreglar problemas en la implementación. Éstas inluyen el uso de valgrind (memcheck) para detectar y arreglar leaks de memoria.

CP-01

Descripción: Se prueba la captura con fsni-src-fuse en una jaula chroot con root en punto de montaje del sistema de ficheros: ejecución de test.c y firefox.

Entrada: Se ejecutará:

```
$ fsni-src-fuse --output=cp01.log
 -o allow_root --with-buffer ~/mnt
$ chroot --userspec=user:user ~/mnt
 $CWD/test
$ chroot --userspec=user:user ~/mnt
 firefox
```

Salida: Se espera que:

- test.c genere la salida esperada
- firefox pueda ejecutarse normalmente

Entorno:
- Módulo fuse insertado en el kernel
- El directorio ∼/mnt está vacio

CP-02

Descripción: Se prueba la captura con fsni-src-pt: ejecución de test.c y firefox.

Entrada: Se ejecutará:

```
$ fsni-src-pt --output=cp02a.log
--with-buffer ./test
$ fsni-src-pt --output=cp02b.log
--with-buffer firefox
```

Salida: Se espera que:

- test.c genere la salida esperada
- firefox pueda ejecutarse normalmente

Entorno: n/a

CP-03

Descripción: Se prueba la captura con fsni-src-klinux: dos solicitudes: test.c, firefox.

Entrada: Se ejecutará:

```
$ fsni-src-kctl
--output=cp03a.log --comm=test
--with-buffer
$ fsni-src-kctl
--output=cp03b.log --comm=firefox
--with-buffer
$ ./test
$ firefox
```

Salida: Se espera que:

- la primera solicitud genere la salida esperada para test.c
- firefox se ejecute sin pérdida perceptible de rendimiento

Entorno:
- Módulo fsni-src-klinux insertado: parámetros por default

CP-04

Descripción: Se prueba la captura con fsni-src-klinux: solicitud de captura a todos los procesos durante 20 minutos.
El usuario debe estresar el sistema durante la prueba.

Entrada: Se ejecutará:

```
$ fsni-src-kctl
--output=cp04.log --with-buffer
$ # estrese el sistema
```

Salida: Se espera que:

- se obtenga un log pesado
- el sistema permanezca estable
- no haya un incremento perceptible de latencia

Entorno:
- Módulo fsni-src-klinux insertado: parámetros por default

CP-05

Descripción: Se prueba el módulo fsni-src-kiocalsa para captura de ioctls `SNDRV_PCM_IOCTL_WRITEI_FRAMES` (ALSA).

Entrada: Se reproduce audio usando ALSA:

```
$ fsni-src-kctl --comm=mpg123
--output=cp05.log --allow=/dev
--policy=deny
$ mpg123 -a hw:0,0
~/file.mp3
```

Salida: Se esperan mensajes IOCTL_DATA después de IOCTL.

Entorno:
- Módulos fsni-src-klinux y fsni-src-kiocalsa insertados: trace_nonregular=1

CP-06

Descripción: Se prueba que fsni-sink-dump puede volcar información de todos los mensajes, basado en la salida de CP-03.

Entrada: Se ejecuta:

```
$ # ejecute CP-03 primero
$ fsni-sink-dump --dump-payload
< cp03.log
```

Salida: Se espera una representación en texto plano del log y volcado hexadecimal+ASCII del payload.

Entorno: n/a

CP-07

Descripción: Se prueba que fsni-sink-gui puede cargar logs pesados, basado en la salida de CP-04.

Entrada:
- Captura → Abrir...
- Cargar el log cp04.log.

Salida: Se espera que el log pueda ser cargado sin errores.

Entorno: n/a

CP-08

Descripción: Se prueba la captura desde fsni-sink-gui: captura con fsni-src-klinux.

Entrada: Ejecución de CP-03(b), esta vez desde fsni-sink-gui.
- Captura → Nueva...
- Introducir el comando usado en CP-03(b)
- Click en Aceptar

Salida: Se espera que fsni-sink-gui actualize la lista de mensajes durante la captura.

Entorno: n/a

CP-09

Descripción: Se prueba que fsni-sink-streams provea la lista de ficheros accedidos.

Entrada: Prueba basada en CP-04, se ejecuta:

```
$ # ejecute CP-04 primero
$ fsni-sink-streams < cp04.log
```

Salida: Se espera lista de accesos durante CP-04.

Entorno: n/a

CP-10

Descripción: Se prueba que fsni-sink-streams puede extraer un stream tipo ioctl.

Entrada: Prueba basada en CP-05, se ejecuta:

```
$ fsni-sink-streams < cp05.log
$ fsni-sink-streams
--path=/dev/snd/pcmC0D0p
--extract=IOC --ioctl=...
| alsaplay
```

Salida: Se espera que el audio capturado sea reproducido usando alsaplay.

Entorno: n/a

CP-11

Descripción: Se prueba que fsni-sink-graph genera un grafo DOT para el caso CP-03(b).

Entrada: Prueba basada en CP-03, se ejecuta

```
$ fsni-sink-graph | neato
 -Tpng:cairo:gd
< cp03a.log > cp03a.png
$ fsni-sink-graph < cp03b.log
> cp03b.dot
```

Salida: Se espera un grafo del estado de cp03b.log

Entorno: n/a

CP-12

Descripción: Se prueba fsni-sink-gs/Gephi, basado en CP-04.

Entrada: Prueba basada en CP-04, con tubería a fsni-sink-gs:

```
$ fsni-src-kctl --output=-
| fsni-sink-gs --bind-addr
127.0.0.1 --bind-port 2480
$ # estrese el sistema
```

Salida: Se espera visualización en tiempo real del grafo para CP-04.

Entorno:

- Gephi está ejecutándose antes de hacer la captura.

CP-13

Descripción:	Se prueba la monitorización de fsni-src-klinux usando procfs.

Entrada:	Se ejecuta:
	`$ # ejecute CP-04`
	`$ cat /proc/fsni-src-klinux`

Salida:	Se espera obtener el estado actual del módulo de kernel.
Entorno:	▪ La prueba CP-04 debe estar en ejecución.

5.2. Evaluación

En esta sección se analiza el rendimiento de los métodos de captura incluidos. La medida utilizada será el tiempo de ejecución para varias copias de datos usando el comando `dd`[2]. Se probará el rendimiento de lectura/escritura por separado[3] ejecutando:

Escritura 400MiB con bs=1M hacia /tmp/.perftest:

```
DD=dd if=/dev/zero of=/tmp/.perftest bs=1M count=400
```

Lectura 400MiB con bs=1M desde /tmp/.perftest:

```
DD=dd if=/tmp/.perftest of=/dev/null bs=1M count=400
```

Las pruebas se ejecutarán para cada caso de la lista siguiente (como el overhead puede ser diferente si se copia el buffer de una lectura/escritura, se incluirán los dos casos):

No trazada Ejecución no trazada (referencia para el resto).

fsni-src-fuse Ejecución con FUSE+chroot:

```
$ fsni-src-fuse --output=- -o allow_root --allow=/tmp/.perftest --policy=deny
~/mnt > /dev/null
$ sudo chroot --userspec=user:user ~/mnt $DD
```

fsni-src-fuse –with-buffer Ejecución FUSE+chroot con –with-buffer:

```
$ fsni-src-fuse --output=- -o allow_root --allow=/tmp/.perftest --policy=deny
--with-buffer ~/mnt > /dev/null
$ sudo chroot --userspec=user:user ~/mnt $DD
```

fsni-src-pt Ejecución ptrace:

```
$ fsni-src-pt --output=- --allow=/tmp/.perftest --policy=deny $DD > /dev/null
```

fsni-src-pt –with-buffer Ejecución ptrace con –with-buffer:

```
$ fsni-src-pt --output=- --allow=/tmp/.perftest --policy=deny
--with-buffer $DD > /dev/null
```

fsni-src-klinux Ejecución trazada con lkm y –comm=dd:

```
$ fsni-src-kctl --output=- --comm=dd --allow=/tmp/.perftest --policy=deny > /dev/null
```

fsni-src-klinux –with-buffer Ejecución trazada con lkm y –comm=dd –with-buffer:

[2] Aunque otros benchmarks (IOzone, Bonnie++...) podrían haberse usado.
[3] Para esto será importante pasar opciones –allow/–deny en la captura.

```
$ fsni-src-kctl --output=- --comm=dd --allow=/tmp/.perftest --policy=deny
--with-buffer > /dev/null
```

Note que:

- Se redirigió la salida de log a `/dev/null` para evitar el overhead de la escritura en disco.
- Los comandos que usan –with-buffer truncan el buffer capturado a 8KiB por default.
- Debido a que la métrica usada puede ser alterada por motivos ajenos a la traza, antes de ejecutar `dd` se asegurará de que:
  1. No haya procesos esperando tiempo de CPU (carga media ≃0.00) ni carga E/S.
  2. El kernel no tiene contenido cacheado en la caché de páginas:

     ```
     $ echo 3 > /proc/sys/vm/drop_caches
     ```

La tabla 5.1 contiene los resultados de la prueba. La figura 5.2 compara los tiempos de ejecución (mejor si más cercano a “No trazada”).

Caso	Lectura (s)	Escritura (s)
No trazada	0.081	0.140
fsni-src-fuse	0.608	0.914
fsni-src-fuse –with-buffer	0.744	0.940
fsni-src-pt	0.094	0.156
fsni-src-pt –with-bufer	0.214	0.347
fsni-src-klinux	0.081	0.139
fsni-src-klinux –with-buffer	0.085	0.142

Tabla 5.1: Comparación de rendimiento

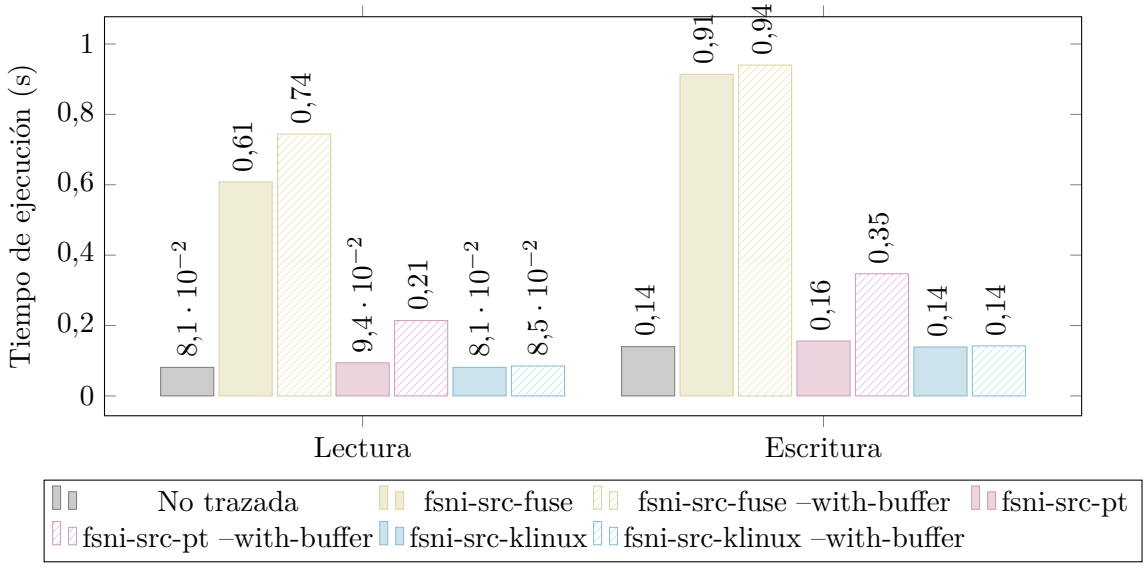


Figura 5.2: Comparación de tiempos de ejecución

Capítulo 6

Planificación

Este capítulo incluye las tareas que han sido parte del desarrollo del proyecto y su planificación temporal. Algunas tareas pueden contener subtareas.

La sección 6.1 describe las tareas del proyecto.
La sección 6.2 incluye un diagrama Gantt de la planificación temporal.

6.1. Tareas

El desarrollo de este proyecto se dividió en las tareas siguientes:

Propuesta de proyecto descripción genérica del proyecto a desarrollar. Descrito por Alejandro Calderón Mateos y Javier Fernández Muñoz.

Análisis análisis de viabilidad y de requisitos:

- Análisis de requisitos de usuario
- Lectura de referencias bibliográficas
- Pruebas de concepto
- Especificación de requisitos software

Diseño del software basado en la especificación de requisitos software

Iteraciones SCRUM de 7 días de duración. Implementación de características adicionales[1] y corrección de errores en cada iteración:

Iteración 1 GNU autotools, libfsni, filters, fsni-src-fuse.

Iteración 2 fsni-sink-dump, fsni-sink-gui (construcción GUI con Glade, carga de logs).

Iteración 3 fsni-sink-gui (captura nueva, guardar log).

Iteración 4 fsni-src-pt.

Iteración 5 fsni-src-klinux (dispositivo de caracteres, regla udev, dirección `sys_call_table`).

Iteración 6 fsni-src-klinux (syscall hooks), fsni-src-kctl.

Iteración 7 fsni-src-klinux (multicast).

[1] Ordenadas por prioridad de la especificación de requisitos software,excepto dependencias.

Iteración 8 fsni-src-klinux (KOH).

Iteración 9 fsni-src-kiocalsa, fsni-sink-streams.

Iteración 10 fsni-sink-graph.

Iteración 11 fsni-sink-gs.

Pruebas (+ depuración) que aseguran el funcionamiento del software.

Evaluación de rendimiento de los métodos de captura.

Documentación generar documentación del proyecto:

- Documentación distribuible: incluida en el directorio `doc/`.
- Paquetes LaTeX: escribir paquetes `.sty` que se usarán en el documento.
- Figuras Ti*k*Z/PGF que se insertarán en la memoria.
- Memoria: éste documento.
- Presentación de diapositivas que acompaña a la memoria.

6.2. Planificación temporal

El proyecto ha tenido una duración de 13 meses y 2 semanas, iniciado el 16 de mayo de 2014 y finalizado el 30 de junio de 2015. La figura 6.1 contiene un diagrama Gantt para este proyecto.

Para información acerca del coste del proyecto, vea el capítulo 7.

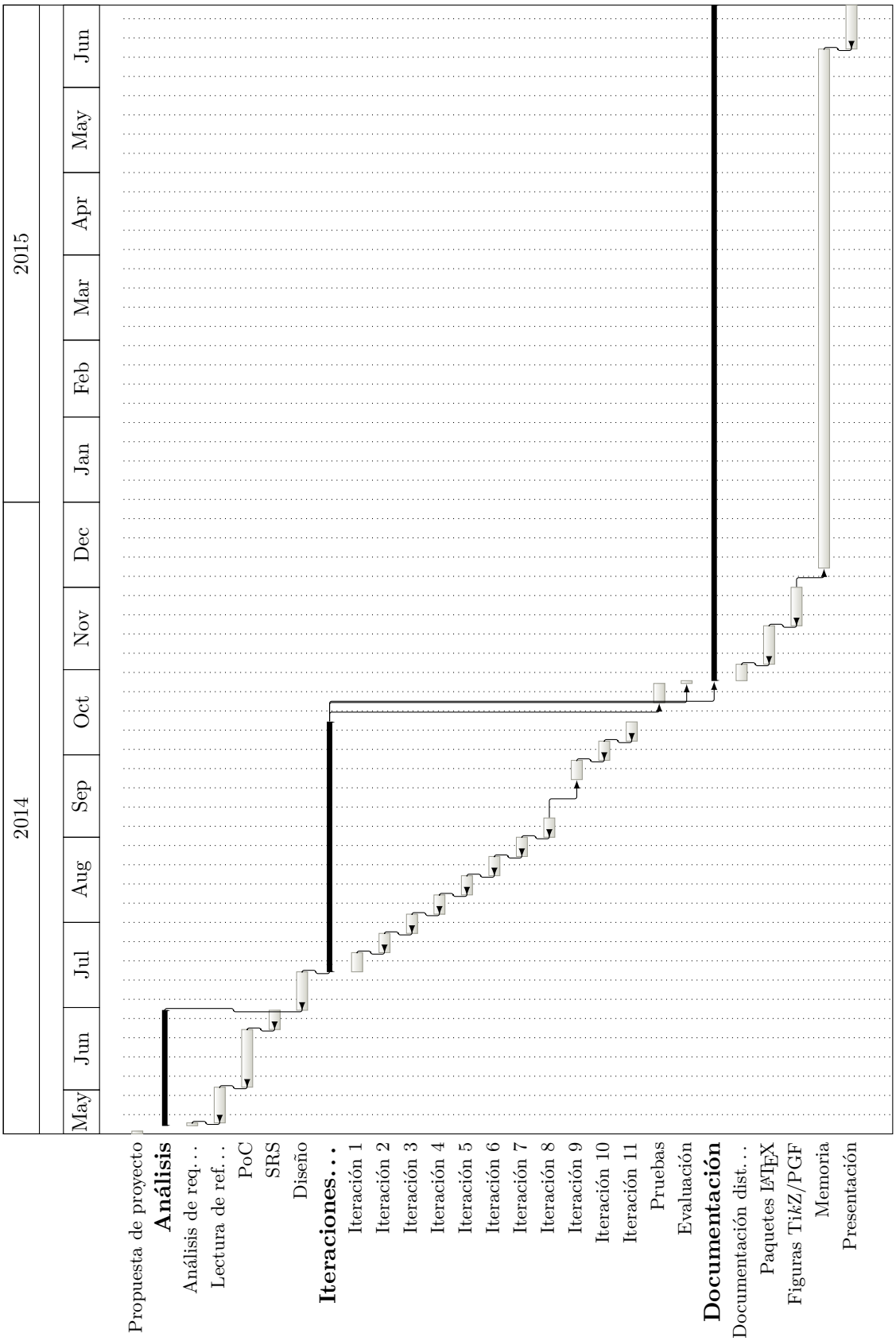


Figura 6.1: Diagrama Gantt del proyecto

Capítulo 7

Presupuesto

En esta sección se presenta el presupuesto requerido para llevar a cabo este proyecto. La duración total del proyecto ha sido de 13 meses y medio, teniendo en cuenta una dedicación de 40 horas semanales con 1 mes de vacaciones.

Los costes no incluyen el 21 % de I.V.A., que será añadido en la sección 7.6.

7.1. Desglose de tareas del proyecto

El cálculo total de horas ha tenido en cuenta todas las tareas descritas en el capítulo 6. En la tabla 7.1 puede apreciarse el total de horas para cada grupo de tareas.

Tarea	Horas
Análisis y diseño	320 h
Implementación	440 h
Pruebas	48 h
Documentacion	1368 h

Tabla 7.1: Desglose de horas por tarea

7.2. Gasto en personal imputable al proyecto

El proyecto ha sido ejecutado por un ingeniero cuyo coste/hora es de 45,00€. El coste total de personal es de 97.920,00€ como puede apreciarse en la tabla 7.2.

Cargo	Horas	Coste/Hora	Total
Ingeniero	2176 h	45,00€/hora	97.920,00€
Total	2176 h		97.920,00€

Tabla 7.2: Coste de personal

7.3. Recursos materiales empleados

Durante la realización de este proyecto se ha hecho uso de software libre y gratuito, por lo que el coste repercutido al cliente es de 0€.

7.4. Amortizaciones

En esta sección se incluyen los costes derivados de la pérdida de valor de un equipo por su uso y que serán repercutidos en el cliente. Para su cómputo se ha considerado que un mes tiene 160 horas laborales (4 semanas de 40 horas).

Éstos pueden verse en la tabla 7.3.

Recurso	Precio	Amortización (meses)	Uso	Repercutido
Portátil	1.400,00€	60	50 %	158,66€
Sobremesa	1.200,00€	60	50 %	136,00€
Total				294,66€

Tabla 7.3: Coste por amortizaciones

7.5. Gastos indirectos

Aquí se tiene en cuenta el coste de todos aquellos elementos que, si bien no repercuten directamente sobre el proyecto, suponen un gasto constante durante la vida de éste.

El material de oficina incluye folios, un bolígrafo y tóner. Para el gasto en electricidad, se ha supuesto un coste de 0,13€ por kWh, con un consumo medio de 90W a la hora.

En la tabla 7.4 hay un resumen de los gastos indirectos del proyecto.

Descripción	Coste
Material de oficina	80,00€
Electricidad	25,46€
Total	105,36€

Tabla 7.4: Costes indirectos

7.6. Resumen del presupuesto

Con los datos de las secciones anteriores se procede a calcular el precio de venta de la solución desarrollada. Se incluye un resumen del presupuesto en la tabla 7.5.

Para compensar posibles imprevistos (roturas de material, enfermedad del personal, etc.) se aplicará un margen del 10 % sobre el precio. El margen de beneficio a aplicar es del 25 % sobre el precio con imprevistos.

El precio final de la solución, contando el 21 % de I.V.A. es **163.579,93€**.

Descripción	Coste
Personal	97.920,00€
Material	0,00€
Amortizaciones	294,66€
Indirectos	105,36€
Margen de imprevistos (10 %)	9.832,00€
Margen de beneficio (25 %)	27.038,00€
21 % I.V.A.	28.389,91€
Total	163.579,93€

Tabla 7.5: Resumen del presupuesto

Capítulo 8

Trabajos futuros

Tras terminar el proyecto hay muchas mejoras que sería deseable poder añadir. A continuación puede verse una lista de aquellas que se han considerado más importantes para continuar este proyecto:

Soportar otras arquitecturas/sistemas operativos en fsni-src-pt. La interfaz de la llamada a sistema `ptrace()` varía entre sistemas operativos. Para complicar más las cosas, el paso de argumentos en una llamada a sistema es dependiente de la arquitectura.

Aún así es posible extender la clase pt_syscall con más `#ifdefs` para dar soporte a estos casos.

Manejo de operaciones de inodo. Hasta este momento se hookean las tablas `file_operations`, `vm_operations_struct` y `address_space_operations`. Esto provee una vista razonablemente completa de lo que está sucendiendo, aunque sería desable emitir mensajes para algunas operaciones de inodo (truncate(), etc.).

Esto requiere cambios en libfsni para soportar nuevos tipos de mensaje.

Módulos para otros kernels. fsni-src-klinux es una alternativa de bajo *overhead* pero sólo puede usarse en un kernel Linux.

Aunque el esfuerzo es considerable, podrían escribirse implementaciones para otros kernels (FreeBSD, etc.).

Proveer algoritmos para encontrar sys_call_table en otras arquitecturas. En este momento fsni-src-klinux sólo provee algorimos para encontrar este símoblo en x86 y amd64. Para otras arquitecturas, root debe obtener manualmente esta dirección desde System.map. Además, el módulo no ha sido probado en otras arquitecturas y puede ser necesario ajustar algo de código.

Ésto permitiría que root no tuviese que preocuparse hacer `grep` en System.map.

Reciclaje de grupos multicast. En estos momentos el módulo de kernel crea un nuevo grupo multicast para cada fichero que vaya a ser trazado, incluso si ya existe un grupo multicast con los mismos miembros.

Reciclar los grupos existentes permite un ahorro de memoria.

Alternativa para tee()/splice(). El objetivo de este cambio es permitir la captura desde fsni-sink-gui en sistemas POSIX que no dispongan de estas llamadas a sistema.

Podrían simularse éstas con `read()`/`write()`, aunque con el *overhead* de una copia a espacio usuario.

Mejora de fsni-sink-gui. En estos momentos éste provee volcado básico de logs. Se podrían añadir las mejoras siguientes:

- El widget GtkTreeView está usando un modelo GtkListStore que tiende a ser lento. Esto puede solucionarse escribiendo un modelo personalizado para GtkTreeView.
- Referenciar mensajes relacionados, e.g. en un mensaje RW, referenciar el mensaje OPEN que abrió el fichero.
- *Plugins* que añadan características, e.g. proveer cadenas legibles para el argumento *cmd* de IOCTL (`cmd=FS_IOC_GETFLAGS` en lugar de `cmd=0x80086601`).
- Implementar filtro de visualización (ver el proyecto wireshark).

Incluir páginas de manual. Escribir manuales de usuario que se instalarán con el software y que el usuario pueda consultar con las ordenas `man` o `info`. Para esta mejora se hará uso de GNU texinfo.

Escribir módulos fsni-src-kliunx-ioc adicionales. Este proyecto incluye el módulo fsni-src-kiocalsa que maneja el ioctl `SNDRV_PCM_IOCTL_WRITEI_FRAMES` (escritura de frames entrelazados ALSA) generando un mensaje IOCTL_DATA.

Otros drivers (tty, sg...) aceptan ioctls que pueden ser de interés (ver `Documentation/ioctl/ioctl-number.txt` de linux).

Bindings libfsni para otros lenguajes. El objetivo de esta mejora es generar bindings para otros lenguajes de programación (python, perl, etc.), de modo que puedan leerse logs desde éstos.

Capítulo 9

Conclusiones

Las conclusiones se han dividido en producto (consecución de objetivos), proceso (problemas en la planificación) y personales.

Este software es GPL, la sección 9.4 incluye las condiciones bajo las que se libera el código.

9.1. Producto

- Se han provisto diferentes mecanismos de captura que no requieren instrumentación de programa ni de kernel.
- Se incluyen herramientas de análisis que permiten volcar un log, listar los ficheros que han sido accedidos, extraer un stream y visualización de grafos. Las utilidades interoperan con todos los origenes usando la capa libfsni.
- El módulo de kernel implementado ha hecho uso de técnicas usadas en rootkits, tales como hook de llamadas a sistema y hook de objetos de kernel (KOH). Para esto se ha tenido que encontrar la dirección del símbolo `sys_call_table` (no exportado por el kernel) y hacer escribible esa página.
- Se puede trazar con bajo overhead con el módulo de kernel incluido; el buffer circular evita contener al consumidor mientras el productor está escribiendo.
- Por último, se aporta a la comunidad una herramienta para el análisis E/S. Para más información, vea más abajo.

9.2. Proceso

- La implementación ha requerido más tiempo del planificado debido a la dificultad para depurar el código que corre en espacio kernel. Además, parchear estructuras del kernel mientras está corriendo ha traído más problemas.

9.3. Personales

- Algunos contenidos incluídos en las asignaturas de Ingeniería del Software y Sistemas Operativos han sido útiles para este proyecto.

- Se ha aprendido sobre el kernel Linux y su interfaz con espacio usuario en la arquitectura Intel x86.
- Se ha aprendido GTK+/GDK para la implementación de fsni-sink-gui.
- Para la escritura de este documento se han escrito algunos paquetes LaTeX 2_ε que han requerido aprender de TeX y Lua.

9.4. Código abierto

El código del proyecto se libera bajo las siguientes condiciones:

Código fuente liberado bajo la licencia GNU General Public License (ver sección E.1).

Documentación liberado bajo la licencia GNU Free Documentation License (ver sección E.2).

Paquetes LaTeX los paquetes `tikzswlayers.sty` (usado para hacer la figura 4.6 y otras; ver listado 9.1), `softwareengineering.sty` (plantillas para ingeniería del software y generar matrices de trazabilidad; ver listado 9.2), `documentation.sty` (dependencia de `softwareengineering.sty`) y `thesis.sty` se liberan bajo GNU General Public License y se subirán a CTAN (Comprehensive TeX Archive Network).

Listado 9.1: Código para generar la figura 4.6

```
\begin{tikzpicture}[every node/.style={font=\footnotesize}]
  \node(c1)[container={4.4cm}{4.9cm}{cat /tmp/d05URoIT},container l1];
  \node(c1n)[rect,adjust to container=c1,at container bottom=c1]{glibc};

  \node(c2)[container={7.2cm}{4.9cm}{fsni-src-pt},container l1,right=\containermargin of
      c1];
  \matrix(m)[matrix of rect,matrix adjust to container={c2}{1},at container bottom=c2]{
    pt\_syscall\_log   \\
    pt\_syscall\_fdvma \\
    pt\_syscall        \\
    glibc              \\
  };

  \userkernelsep(uks){c1.south west}{c2.south east};

  \node[rect,sys component,xadjust to={c1n.west}{m-3-1.east}{0pt},%
        below userkernelsep=uks]{syscall interface};

  \begin{scope}[execution path]
    \draw (1   ,1.5)-- (1   ,-1)  node[near start]{2} --
          (1.5 ,-1) -- (1.5 ,1.5) node[near start]{8};
    \draw (5.5 ,2)  -- (5.5 ,-1)  node[near start]{1} --
          (6   ,-1) -- (6   ,4)   node[midway]{3} node[at end]{4} --
          (6.5 ,4)  -- (6.5 ,-1)  node[near start]{5} --
          (7   ,-1) -- (7   ,4)   node[near end]{6} node[very near end]{7};
  \end{scope}
\end{tikzpicture}
```

Listado 9.2: Código para generar las tablas 3.1y 3.2

```
\foreach \opts [count=\this] in {{e_offset=0,e_count=24},%
                                 {e_offset=24}} {
```

```
3     \begin{table}[!ht]
4     \centering\traceabilityPrintMatrix{F\37-SR}{CA\37-UR}{\opts}
5     \caption{Matriz de trazabilidad F-SR/CA-UR (\this\space de 2)}\label{tab:trace_fsr_caur
          _\this}
6     \end{table}
7   }
```

Apéndice A

Referencia de libfsni

A.1. libfsni/libfsni.h

En este fichero de cabecera se definen los tipos y macros de las que depende el resto de libfsni.

El desplazamiento del miembro `union __msg_any` no es fijo y depende de si el mensaje incluye un `struct msghdr_src`. Es por esto que para obtener acceso a los miembros `struct msghdr_source` y `union __msg_any` simpre debería usar las macros definidas que se detallan en estas subsecciones.

A.1.1. fsni_get_msghdr_src

`#define fsni_get_msghdr_src(_m)` [Macro]

Devuelve un puntero al `struct msghdr_src` de $\_m$ o `NULL` si el mensaje no tiene origen (flag `F_GROUP` activo).

A.1.2. fsni_get_msg_any

`#define get_msg_any(_m, _t)` [Macro]

Devuelve un tipo $\_t$ para los datos dependientes del mensaje $\_m$.

```
struct __msg_open *mo = fsni\_get_msg_any(&m, struct __msg_open *);
```

A.2. libfsni/libfsni_write.h

La interfaz que libfsni provee para serialización de mensajes es simple. Las siguientes subsecciones son referencia de las funciones/macros definidas.

Para usar esto debe incluir:

```
1 #include <libfsni/libfsni.h>
2 #include <libfsni/libfsni_write.h>
```

libfsni requiere que el productor implemente las rutinas que escriben un buffer. Vea la sección A.3 para más información.

A.2.1. fsni_msghdr_init_group

`void fsni_msghdr_init_group(struct msghdr *m, __u8 type, __u16 flags, __u32 seq);` [Función]

Inicializa la cabecera de *m* y activa el flag `F_GROUP`. Como resultado, el mensaje no incluirá origen.

A.2.2. fsni_msghdr_init

`void fsni_msghdr_init(struct msghdr *m, __u8 type, __u16 flags, __u32 seq, pid_t tgid, pid_t pid, const char *comm);` [Función]

Inicializa un msghdr con información de origen. La cabecera de *m* se inicializa con *type*, *flags* y *seq*. El resto de argumentos (*tgid*, *pid* y *comm*) se usan para inicializar msghdr_src.

```
fsni_msghdr_init(&m, T_OPEN, 0, seq++,
                  current->tgid, current->pid, current->comm);
```

A.2.3. fsni_set_msg_any

`#define fsni_set_msg_any(_m, _t, ...)` [Macro]

Establece la información dependiente del mensaje. *m* es un puntero al mensaje; *_t* es el tipo del mensaje. El resto de argumentos se usan para inicializar el tipo *_t*:

```
feni_set_msg_any(&m, struct msg_open, filp, flags, mode);
```

A.2.4. fsni_set_msg_any_err

`#define fsni_set_msg_any_err(_m, _err, _t, ...)` [Macro]

Como *fsni_set_msg_any* pero además establece *_err* como valor de err en el mensaje.

A.2.5. fsni_msghdr_setpayload_fn

`void fsni_msghdr_setpayload_fn(struct msghdr *m, const void *payload, int _len, int _max, write_payload_t fn)` [Función]

Establece el payload para un mensaje especificando la función *fn* que se usará para escribirlo. *_len* especifica la longitud del payload y será truncado a *_max* bytes. En este caso, se activará el flag `F_TRUNCATED`.

A.2.6. fsni_msghdr_setpayload

`void fsni_msghdr_setpayload(struct msghdr *m, const void *payload, int _len, int _max)` [Función]

Como *fsni_msghdr_seypayload_fn* pero usa `__write_local` para escribir el payload.

A.2.7. fsni_write_msghdr

`void fsni_msghdr_write(struct msghdr *m, void *arg)` [Función]

Escribe el mensaje *m* invocando las funciones de escritura de bajo nivel. Se usará `__write_local` para escribir la cabecera y se invocará a la función correspondiente para escribir el payload. *arg* se pasa tal cual a estas funciones.

Si el destino de la escritura tiene una capacidad limitada, tal como un buffer circular, puede usar `m->length` para determinar la longitud del mensaje antes de escribirlo.

Habitualmente *arg* es usado por la implementación como destino del mensaje.

A.3. Funciones de bajo nivel

libfsni requiere que el productor implemente al menos la función `__write_local()`. Ésta se invoca desde `fsni_write_msghdr()` para escribir la cabecera y el payload de los mensajes que no especifican una función personalizada. De esta manera, el usuario tiene el control de como los mensajes son escritos i.e. `write()` sobre un descriptor de fichero, copiados sobre un buffer anillo...

El prototipo de todas las funciones de bajo nivel es[1]:

```
int write_function_proto(void *arg, const char *buffer, int len);
```

El implementador debería escribir *len* bytes del buffer apuntado por *buf*. El argumento *arg* es pasado tal cual desde `fsni_write_msghdr()`.

Una implementación sencilla en la que arg es un `FILE *` podría ser así:

```
1 int __write_local(void *arg, const char *buffer, int len) {
2         return fwrite(buffer, 1, len, (FILE *)arg);
3 }
```

A.4. libfsni/libfsni_read.h

El consumidor debe incluir:

```
1 #include <libfsni/libfsni.h>
2 #include <libfsni/libfsni_read.h>
```

A.4.1. fsni_open_channel

`struct fsni_channel *fsni_open_channel(FILE *stream);` [Función]

Inicializa un canal para leer desde `stream`. Invoca automáticamente a `fsni_read_hdr()`.

A.4.2. fsni_read_hdr

`int fsni_read_hdr(struct fsni_channel *c);` [Función]

Lee la cabecera global desde `c`. No debería invocar esta función más de una vez.

[1] El tipo write_payload_tes un puntero a una función con este prototipo.

A.4.3. fsni_read_msghdr

`int fsni_read_msghdr(struct fsni_channel *c, struct msghdr *m);` [Función]

Invoque a esta función para obtener el siguiente mensaje desde `c`. El miembro `m->__payload` deberá apuntar a una región de memoria preasignada con suficiente capacidad para copiar el payload del mensaje.

Puede acceder directamente a los miembros de struct msghdr después de invocar esta función.

A.5. Referencia de tipos

[0] OPEN

Flags: `---T-`

Descripción: Un fichero ha sido abierto por `open()` o equivalente.
file se usará en siguientes mensajes y hace referencia a este fichero mientras estéabierto.
flags y *mode* son los argumentos pasados a open().

Estructura:
```
struct __msg_open {
    __u64 file;
    __u32 flags;
    __u16 mode;
} __attribute__((packed));
```

Payload: Path al fichero abierto.

[1] RELEASE

Flags: `-----`

Descripción: El último *fd* que referenciaba un fichero ha sido cerrado por`close()`.
La referencia *file* queda invalidada.

Estructura:
```
struct __msg_release {
    __u64 file;
} __attribute__((packed));
```

Payload:

[2] RW

Flags: `-SWT-`

Descripción: Ha ocurrido una operación de lectura/escritura en una parte de*file*.
buffer es la dirección del buffer en espacio usuario.
len es el número de bytes solicitados.
pos es la posición en elfichero al ejecutar la operación.
ret_len es el numero de bytes transferidos.

Estructura:
```
struct __msg_rw {
    __u64 file;
    __u64 buffer;
    __u64 len;
    __u64 pos;
    __u64 ret_len;
} __attribute__((packed));
```

Payload: Buffer transferido.

[3] RW_ITER

Flags: `ASW--`

Descripción: Ha ocurrido una operación de lectura/escritura en una lista de *iovec* (scatter-gatter) en *file*.
nr_segs es el número de IOVECs pasados.
pos es la posición en el fichero.
ret_len es el número de bytes transferidos.

Estructura:
```
struct __msg_rw_iter {
    __u64 file;
    __u16 nr_segs;
    __u64 pos;
    __u64 ret_len;
} __attribute__((packed));
```

Payload:

[4] IOVEC

Flags `---TG`

Descripción: Un IOVEC (parte de una operación RW_ITER). Como tal, se escribe siempredespués de un mensaje RW_ITER con el flag `G` activo. *iov_base* y *iov_len*es la dirección del buffer y el tamaño del mismo respectivamente.

Estructura:
```
struct __msg_iovec {
    __u64 iov_base;
    __u64 iov_len;
} __attribute__((packed));
```

Payload: Buffer transferido.

[5] IOCTL

Flags: `-----`

Descripción: Un proceso invocó a `ioctl()` sobre *file*.
cmd y*arg* son los argumentos pasados en la llamada ioctl().

Estructura:
```
struct __msg_ioctl {
    __u64 file;
    __u32 cmd;
    __u64 arg;
} __attribute__((packed));
```

Payload:

[6] IOCTL_DATA

Flags: `---TG-`

Descripción: Datos pasados en una llamada ioctl(). Soportado únicamente por fsni-src-klinux+ módulo -kiocxxx.
data_nr identifica estos datos.

Estructura:
```
struct __msg_ioctl_data {
    __u16 did;
} __attribute__((packed));
```

Payload: Dependiente del argumento *cmd* de la llamada ioctl().

[7] MMAP

Flags: `-----`

Descripción: El fichero *file* ha sido mapeado en el espacio de direcciones de unproceso.
vm_start y *vm_end* son las direcciones del nuevo mmap enespacio usuario.
flags del nuevo area de memoria.
pgoff esel desplazamiento en el fichero en unidades PAGE_SIZE.

Estructura:
```
struct __msg_mmap {
    __u64 file;
    __u64 vm_start;
    __u64 vm_end;
    __u32 flags;
    __u64 pgoff;
} __attribute__((packed)):
```

Payload:

[8] MUNMAP

Flags: `-----`

Descripción: Una parte de un mmap ha sido desmapeado del espacio de direcciones.
vm_start y *vm_end* es el intervalo de direcciones que ha sido desmapeado.

Estructura:
```
struct __msg_unmap {
    __u64 vm_start;
    __u64 vm_end;
} __attribute__((packed));
```

Payload:

[9] VM_FAULT

Flags: `--WT-`

Descripción: Ha ocurrido un fallo de página en un area de memoria mapeada a un fichero.`W=0`: lectura de memoria; `W=1`: escritura de memoria.
*flags*del fallo de página.
virtual_address es la dirección en la que se provocóel fallo.
page es un puntero a la página; se referenciará en WRITEPAGE.

Estructura:
```
struct __msg_vm_fault {
    __u32 flags;
    __u64 virtual_address;
    __u64 page;
} __attribute__((packed));
```

Payload: La página que provocó el fallo.

[10] WRITEPAGE

Flags: `--WT-`

Descripción: Una página modificada fue volcada a un dispositivo de bloques. El flag`W` siempre está activo.
page es un puntero a la página.

Estructura:
```
struct __msg_writepage {
    __u64 page;
} __attribute__((packed));
```

Payload: La página que fué volcada a disco.

Apéndice B

Manual de usuario

Este capítulo contiene instrucciones de uso para el usuario, además de capturas de pantalla en las que puede ver el resultado de la ejecución de éstas.

B.1. Captura

En esta sección se describe el procedimiento que se debe seguir para hacer una captura nueva. Normalmente, deberá seguir estos pasos[1]:

1. Seleccionar un SRC[2] (`fsni-src-fuse`, `fsni-src-pt` o `fsni-src-kctl`) (ver tabla B.1 para una comparativa). Probablemente quiera usar `fsni-src-kctl` si el módulo de kernel está disponible en su sistema.

2. Ejecutar el programa sin argumentos para ver ayuda adicional y qué opciones están disponibles. En la figura B.1 puede verse el uso de `fsni-src-kctl`.

3. Ejecutar la linea de comando con argumentos.

Captura de ioctl

La captura de un ioctl es automática si el módulo de kernel requerido está insertado. Si ha localizado el módulo que necesita, solicite a root su inserción.

B.1.1. Captura desde consola

En la figura B.2 puede verse un ejemplo de ejecución desde consola. Para interrumpir la captura envíe una señal SIGINT (`kill -SIGINT` o Ctrl+C).

Estado del módulo

Si usa el módulo de kernel para hacer una captura, puede ver el estado de éste ejecutando:

```
$ cat /proc/fsni-src-klinux
```

[1] Aquí se ha supuesto que usted ya sabe qué proceso trazar.
[2] Recuerde que un SRC inicia una captura y proveeun log de ésta.

Característica	fsni-src-fuse	fsni-src-pt	fsni-src-kctl
Overhead de la captura	Medio	Alto	Bajo
Módulo de kernel no requerido		•	
Captura R/W	•	•	•
Captura R/W vectorizada	•	•	•
Captura AIO	•	—	•
Captura R/W mapeada (`mmap`)	•		•
Diferencia tipo R/W		•	•
Captura IOCTL		•	•
Captura datos IOCTL		—	•

Tabla B.1: Comparativa de SRC

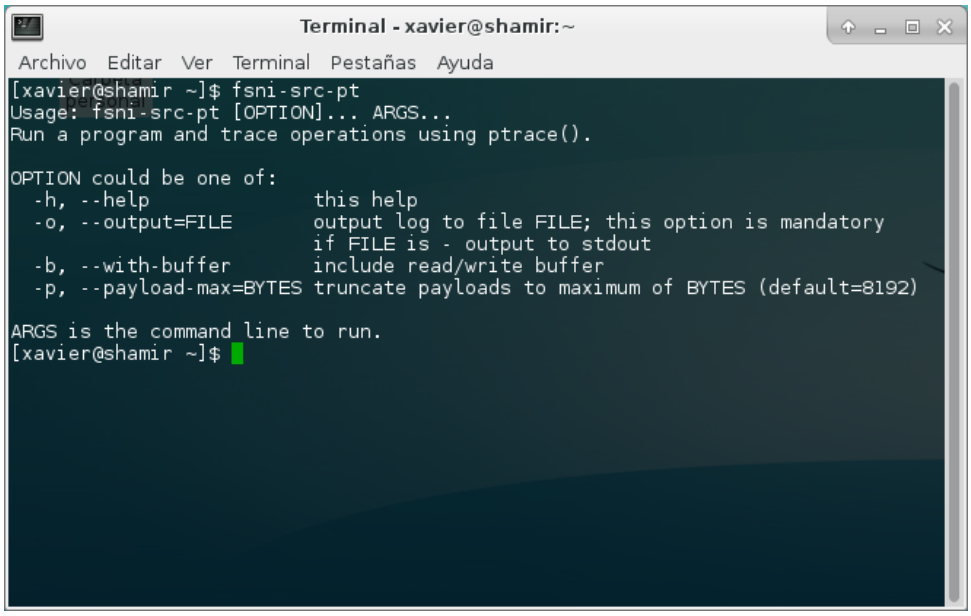


Figura B.1: Uso de fsni-src-pt

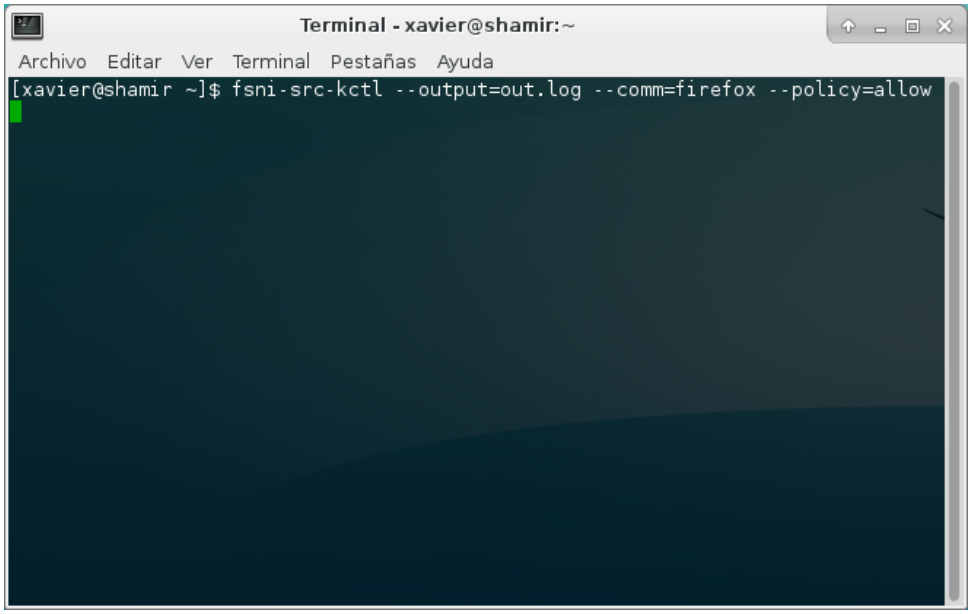


Figura B.2: Ejemplo de captura desde consola

B.1.2. Captura desde GUI

- Ejecute `fsni-sink-gui` en un terminal:

  ```
  $ fsni-sink-gui &
  ```

- Click en Capture → New...
- Escriba el comando con argumentos como lo haría en una consola de texto, asegurándosede que añade la opción `--output=-` (ver figura B.3).
- Click en OK (ver figura B.4); para detener la captura haga click enStop. Puede guardar la captura haciendo click en Capture → Save as...

B.2. Volcar un log

B.2.1. Volcar un log desde consola

Para volcar un log en una consola de texto puede ejecutar:

```
$ fsni-sink-dump < captura.log
```

Puede incluir la opción `-dump-payload` para volcar el payload en hexadecimal+ASCII (ver figura B.5).

B.2.2. Volcar un log desde GUI

Si está ejecutando `fsni-sink-gui`, haga click en Capture → Open...

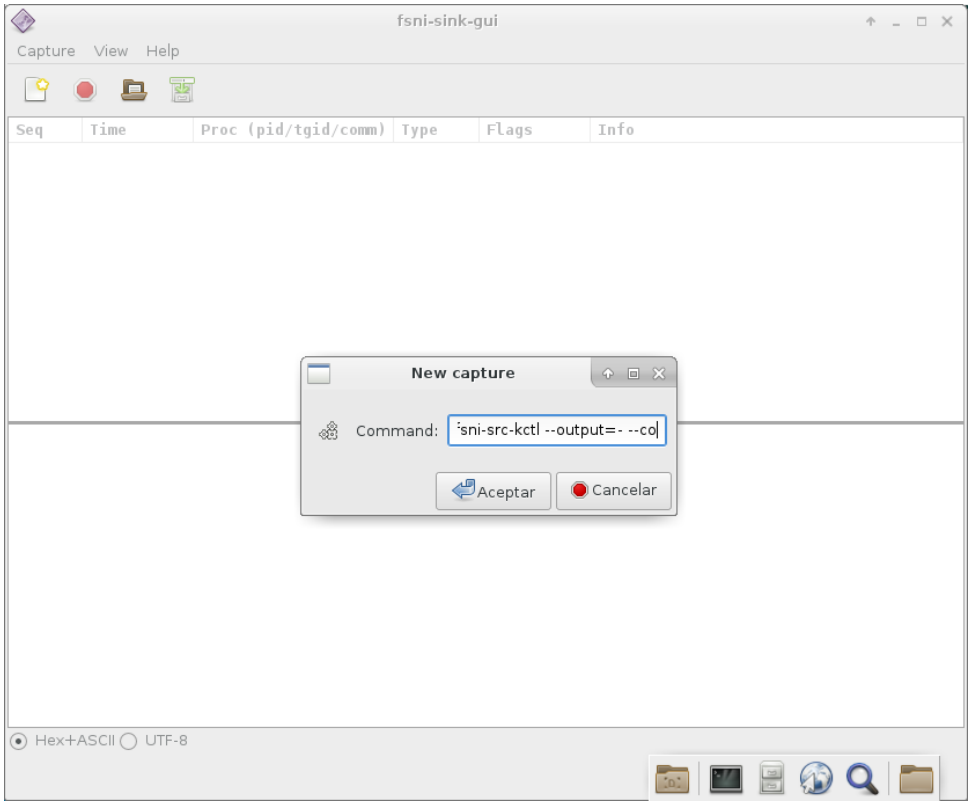


Figura B.3: Captura nueva en fsni-sink-gui

B.3. Manejo de streams de un log

Puede usar la utilidad `fsni-sink-streams` para ver una lista de ficheros accedidos o extraer un stream desde un log:

Para ver la lista de ficheros accedidos, ejecute `fsni-sink-streams` pasando en *stdin* el log (ver figura B.6):

```
$ fsni-sink-streams < captura.log
```

Para extraer un stream ejecute `fsni-sink-streams` con el argumento `-extract` (ver figura B.7). Para más información ejecute:

```
$ fsni-sink-streams --help
```

B.4. Grafos

B.4.1. Generar un grafo

En la figura B.8 puede verse una imágen generada con `fsni-sink-graph` + Gephi. Para ver ayuda adicional ejecute:

```
$ fsni-sink-graph --help
```

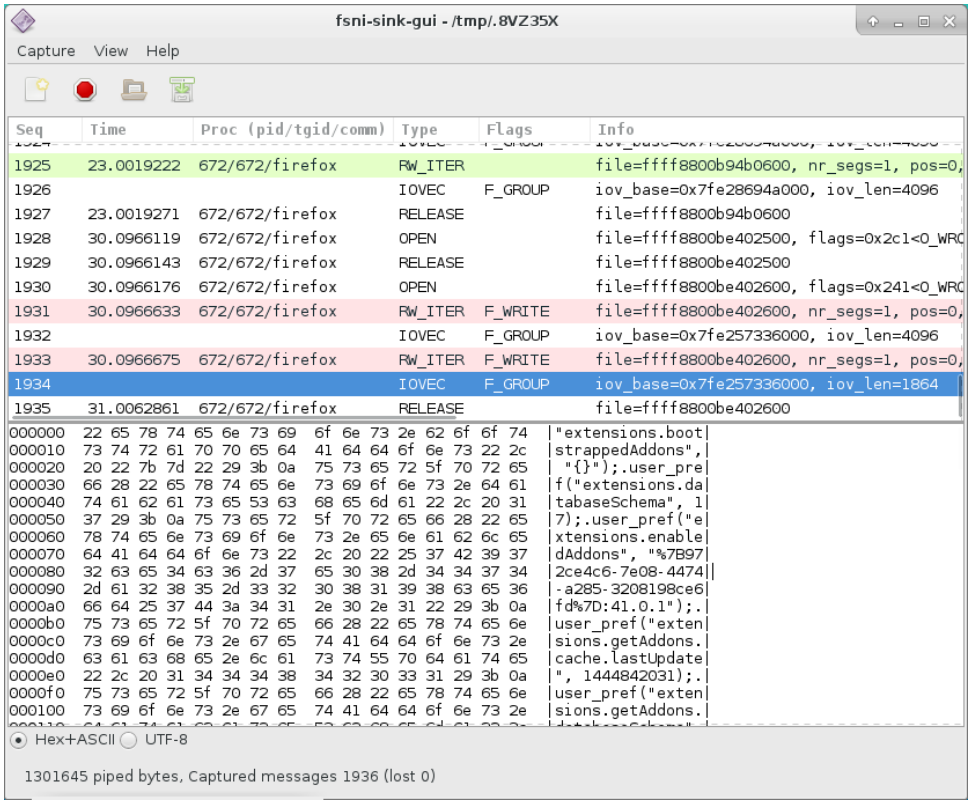


Figura B.4: Captura corriendo en fsni-sink-gui

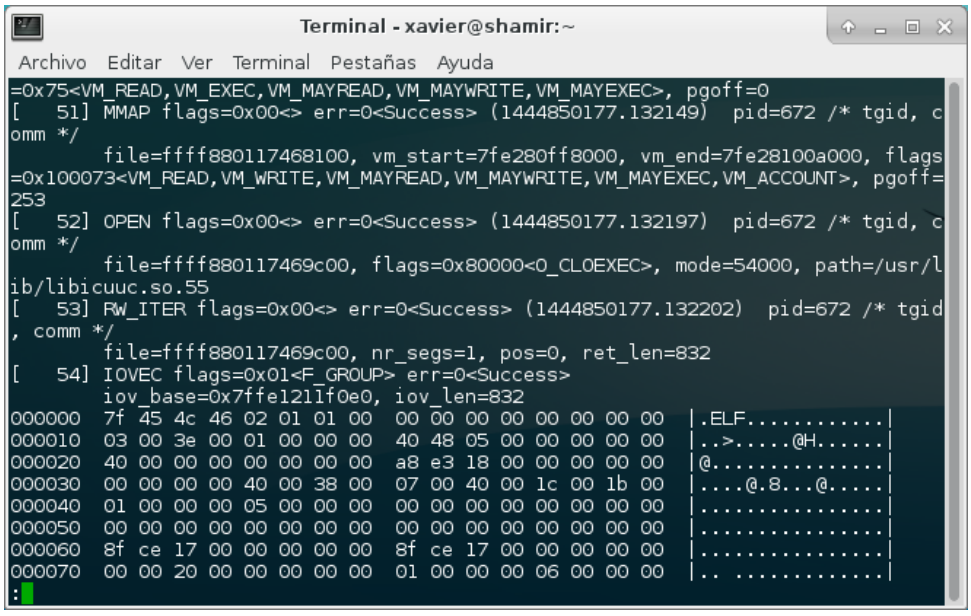


Figura B.5: Salida de fsni-sink-dump

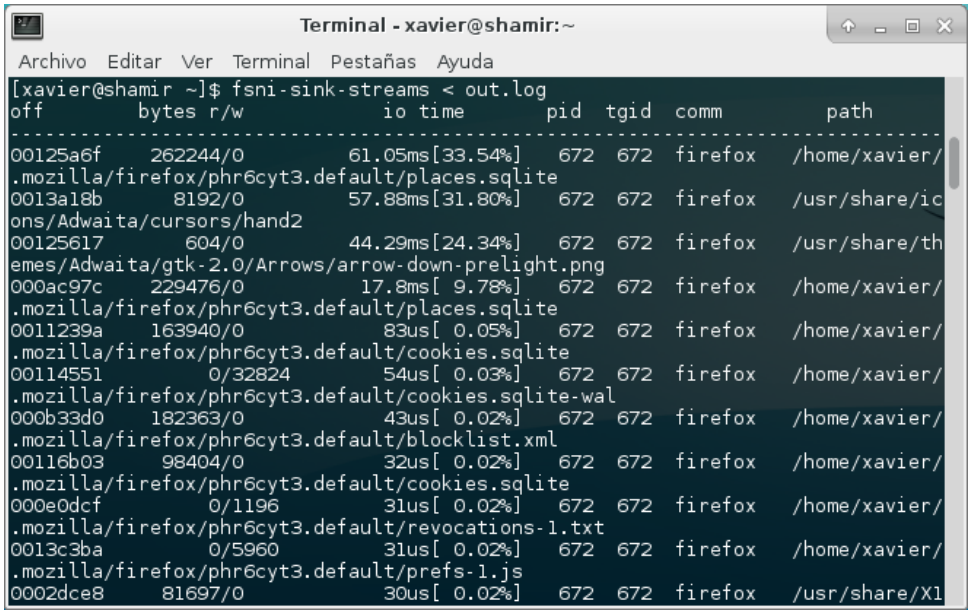


Figura B.6: Lista de ficheros accedidos

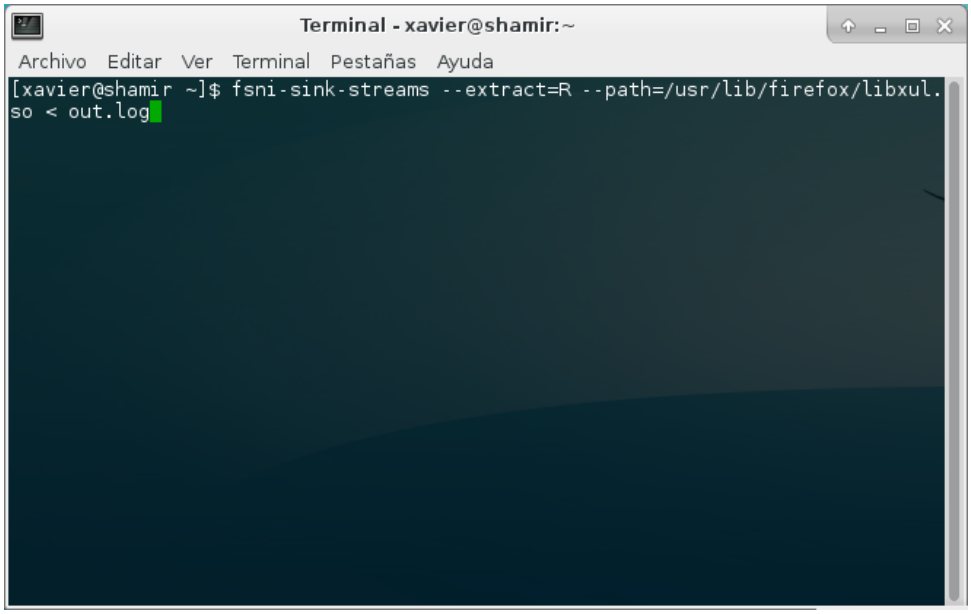


Figura B.7: Extraer un stream con fsni-sink-streams

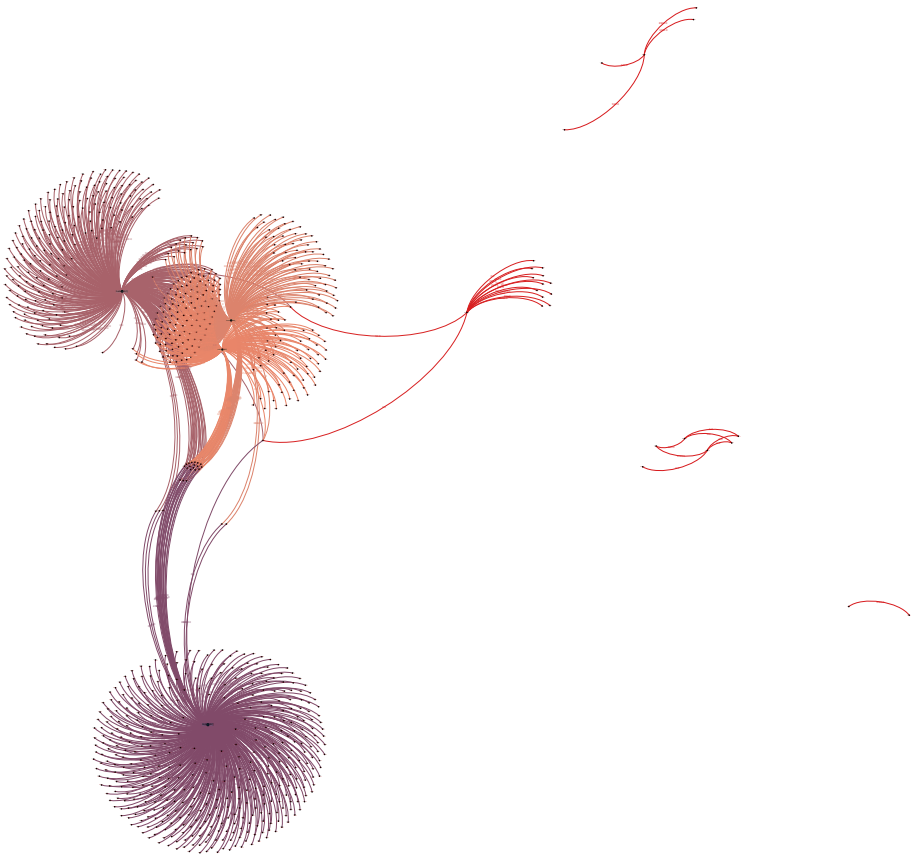


Figura B.8: Imágen generada con fsni-sink-graph + Gephi

B.4.2. Grafo en tiempo real con Gephi

Para esto necesitará Gephi; puede descargarlo en `http://www.gephi.org/`.

Puede pasar un log a `fsni-sink-gs` usando una tubería:

```
$ fsni-src-kctl --output=- --comm=firefox --policy=allo
 | fsni-sink-gs --bind-addr 127.0.0.1 --bind-port 2480
```

Para ver información de uso adicional ejecute:

```
$ fsni-sink-gs --help
```

Instrucciones para Gephi:

- Streaming → click en +
- Source URL: `http://localhost:2480/`
- Click en OK

Apéndice C

Manual de instalación

Este capítulo es una guia para la instalación y administración del software distribuido. La tabla C.1 puede ser utilizada como checklist de la instalación.

C.1. Dependencias

Para compilar y ejecutar el proyecto debe tener instalados los siguientes paquetes adicionales:

- gcc
- autoconf
- automake
- fuse
- linux ≥4.0.9
- linux-headers
- gtk+ ≥3.16

Puede que necesite cargar el módulo fuse durante el arranque del sistema; vea las instrucciones de su distribución para saber cómo hacer esto.

Puede instalar los paquetes requeridos ejecutando el siguiente comando como root (sólo archlinux):

```
# pacman -S gcc autoconf automake fuse linux linux-headers gtk+
```

Hecho	Tarea
	Instalar dependencias (ver sección C.1)
	Descargar una copia del código fuente
	Configurar compilación `./configure`
	Compilar (`make`)
	Instalar (`make install`)
	Configuración post-instalación (ver sección C.3)

Figura C.1: Lista de tareas de la instalación

C.2. Compilación

Puede obtener una copia del repositorio o un tarball generado desde una revisión específica en éste. Si obtiene una copia del repositorio de trabajo, deberá ejecutar `autoreconf` para generar el script `configure` (ya incluido en el tarball).

Para obtener una copia del repositorio deberá ejecutar:

```
$ git clone $LOCATION src
$ cd src/
$ autoreconf -i
```

Alternativamente, para obtener una copia en un tarball:

```
$ wget $LOCATION
$ tar -xzvf src.tar.gz
$ cd src/
```

Ahora ejecute `./configure -help` para ver las opciones disponibles (documentadas en la salida generada):

```
$ ./configure --help
```

Normalmente, puede seguir con las opciones predeterminadas (`-prefix=/usr/local-enable-fuse -enable-pt -enable-klinux -with-gtk -with-udev`). Se recomienda no combinar la opción `-enable-klinux` con `-without-udev`; esto impide que un usuario no privilegiado pueda hacer uso del dispositivo de caracteres.

Para terminar la configuración ejecute `./configure`:

```
$ ./configure
```

Para compilar el software deberá ejecutar `make` ahora:

```
$ make
```

C.2.1. Instalación

Para terminar la instalación procederemos del siguiente modo:

```
$ sudo make install
```

C.3. Administración

C.3.1. Carga del módulo de kernel durante el arranque

Para archlinux, debería bastar con ejecutar el siguiente comando (consulte el manual de su distribución para saber cómo hacerlo en otras distribuciones):

```
$ echo 'fsni-src-klinux' | sudo tee /etc/modules-load.d/fsni-src-klinux.conf
```

Parámetro	Descripción	Default
sys_call_table	Especificar manualmente la dirección del símbolosys_call_table. En x86 o amd64 esto no debería ser necesario.	
allow_max	Especifica el máximo de trazas concurrentes que se permiten.	8
trace_nonregular	Establecer a 1 para trazar ficheros especiales.	0
bufsize_max	Tamaño máximo para un buffer circular (en bytes).	4194304
bufsize_min	Tamaño mínimo (default) para un buffer circular (en bytes).	4194304
null_writepages	Establecer a 1 para hacer que el método writepages deaddress_space_operations sea NULL. Puede usarse cuando laimplementación de writepages del sistema de ficheros no invoca awritepage.	0

Tabla C.1: Parámetros del módulo de kernel

C.3.2. Parámetros del módulo de kernel

El módulo de kernel permite configurar algunos parámetros (ver tabla C.1). Para establecer un parámetro, se puede proceder del siguiente modo:

- Si carga el módulo manualmente, puede pasar el parámetro como argumento a modprobe:

```
$ sudo modprobe fsni-src-klinux PARAM=VALUE
```

- Si carga el módulo automáticamente, añada una linea options a un fichero en /etc/modprobe.d/ (consulte las instrucciones para su distribución):

```
$ echo 'options fsni-src-klinux PARAM=VALUE' \
  | sudo tee /etc/modprobe.d/fsni-src-klinux.conf
```

- Si el módulo ya está insertado, puede ejecutar:

```
$ echo -n 'VALUE' | sudo tee /sys/module/fsni-src-klinux/parameters/PARAM
```

Advertencia

Note que cambiar allow_max o bufsize_max podría ser usado por un usuario para dejar sin memoria el sistema. Hágalo bajo su propia responsabilidad.

C.4. Cancelar traza de un usuario

Para cancelar una traza de otro usuario puede enviar una señal SIGINT a ese proceso (recuerde reemplazar `PID`):

```
$ ps aux | grep src
$ kill -SIGINT PID
```

Apéndice D

Código destacado

En este apéndice se incluye el código fuente considerado de importancia. Por otra parte, la sección D.1 es una guia a la organización de los repositorios usados en el proyecto.

D.1. Organización del código

El proyecto ha sido mantenido en los dos repositorios git siguientes:

src contiene el código fuente del software y la documentación distribuible.

doc contiene esta memoria y la presentación de diapositivas, además de los paquetes LaTeX que se han escrito y las figuras incluidas.

En el repositorio `src` se puede encontrar:

libfsni/ contiene ficheros .c y .h de libfsni

src/ este directorio contiene el código fuente de fsni-{src,sink}-xxx

include/ contiene los ficheros .h para el código en src/

support/ contiene el programa de prueba test.c

En el repositorio `doc` se puede encontrar:

packages/ este directorio contiene los paquetes LaTeX que se han escrito y que se usan en la memoria. Algunas figuras incluidas en images/ requieren el paquete `tikzswlayers.sty`.

shared/ contiene las figuras incluidas en la memoria y en la presentación de diapositivas.

book/ contiene `main.tex` con el esquema de la memoria.

book/content/ contiene el código de los capítulos incluidos en esta memoria.

presentation/ contiene el código de la presentación de diapositivas.

D.2. include/src-klinux/rb_voidp.h

```
1  #ifndef _RB_VOIDP_H
2  #define _RB_VOIDP_H 1
3
4  #include <linux/rbtree.h>
5
6  struct __rb_voidp_ent {
7          struct rb_node node;
8          void *ptr;
9  };
10 #define RB_VOIDP_EMBED(_m)                \
11   union {                                 \
12     struct  __rb_voidp_ent __rb_voidp_ent; \
13     struct {                              \
14       struct rb_node __rb;                \
15       _m;                                 \
16     };                                    \
17   }
18
19 struct __rb_voidp_ent *__rb_voidp_lookup(struct rb_root *root, void *ptr,
20                       struct rb_node **ret_parent, struct rb_node ***rb_link);
21
22 /**
23  * rb_voidp_lookup - lookup @ptr key in @root rbtree
24  */
25 static inline struct __rb_voidp_ent *rb_voidp_lookup(struct rb_root *root,
26                                                      void *ptr) {
27         struct rb_node *parent;
28
29         return __rb_voidp_lookup(root, ptr, &parent, NULL);
30 }
31
32 /**
33  * __rb_voidp_add_hint - add @ent to @root rbtree with node placement hint;
34  * normally used after a call to __rb_voidp_lookup().
35  */
36 static inline void __rb_voidp_add_hint(struct rb_root *root,
37                       struct __rb_voidp_ent *ent, struct rb_node *parent,
38                       struct rb_node **rb_link) {
39         rb_link_node(&ent->node, parent, rb_link);
40         rb_insert_color(&ent->node, root);
41 }
42
43 /**
44  * rb_voidp_add - add @ent to @root rbtree
45  */
46 static inline void rb_voidp_add(struct rb_root *root,
47                                 struct __rb_voidp_ent *ent) {
48         struct rb_node **p, *parent;
49
50         if (!__rb_voidp_lookup(root, ent->ptr, &parent, &p))
51                 __rb_voidp_add_hint(root, ent, parent, p);
52 }
53
54 /**
55  * rb_voidp_delete - delete node with key @ptr
56  */
57 static inline struct __rb_voidp_ent *rb_voidp_delete(struct rb_root *root,
58                                                      void *ptr) {
59         struct __rb_voidp_ent *__ent = rb_voidp_lookup(root, ptr);
60
61         if (__ent)
62                 rb_erase(&__ent->node, root);
63         return __ent;
```

```
64  }
65
66  /**
67   * RB_VOIDP_TYPED - declare rbtree functions for type @_type
68   * @_prefix: function prefix
69   * @_type: arguments/return value of type '_type *'
70   *
71   * Declare functions for manipulation of rbtree of type @_type.
72   */
73  #define RB_VOIDP_TYPED(_prefix, _type)                                      \
74    static inline _type * _prefix ## _lookup(struct rb_root *root, void *ptr) { \
75      struct __rb_voidp_ent *__ent = rb_voidp_lookup(root, ptr);              \
76      return __ent ? container_of(__ent, _type, __rb_voidp_ent) : NULL;       \
77    }                                                                         \
78    static inline _type * _prefix ## _lookup_hint(struct rb_root *root,       \
79                  void *ptr, struct rb_node **p, struct rb_node ***rb_link) {  \
80      struct __rb_voidp_ent *__ent = __rb_voidp_lookup(root, ptr, p, rb_link); \
81      return __ent ? container_of(__ent, _type, __rb_voidp_ent) : NULL;       \
82    }                                                                         \
83    static inline void _prefix ## _add(struct rb_root *root, _type *p) {      \
84      rb_voidp_add(root, &p->__rb_voidp_ent);                                 \
85    }                                                                         \
86    static inline _type * _prefix ## _delete(struct rb_root *root, void *ptr) { \
87      struct __rb_voidp_ent *__ent = rb_voidp_delete(root, ptr);              \
88      return __ent ? container_of(__ent, _type, __rb_voidp_ent) : NULL;       \
89    }
90
91  #endif
```

D.3. src/src-klinux/rb_voidp.c

```
1   #include <rb_voidp.h>
2
3   /**
4    * __rb_voidp_lookup - lookup @ptr in @root rbtree and optionally return node
5    * placement hint; based on cfq-iosched.c: cfq_prio_tree_lookup()
6    */
7   struct __rb_voidp_ent *__rb_voidp_lookup(struct rb_root *root, void *ptr,
8                       struct rb_node **ret_parent, struct rb_node ***rb_link) {
9           struct rb_node **p = &root->rb_node, *parent = NULL;
10          struct __rb_voidp_ent *__ent = NULL;
11
12          while (*p) {
13                  struct rb_node **n;
14
15                  parent = *p;
16                  __ent = rb_entry(parent, struct __rb_voidp_ent, node);
17
18                  if (ptr > __ent->ptr)
19                          n = &(*p)->rb_right;
20                  else if (ptr < __ent->ptr)
21                          n = &(*p)->rb_left;
22                  else
23                          break;
24                  p = n;
25                  __ent = NULL;
26          }
27
28          *ret_parent = parent;
29          if (rb_link)
30                  *rb_link = p;
31          return __ent;
```

```
32  }
```

D.4. include/src-klinux/tcache.h

```
1   #ifndef _TCACHE_H
2   #define _TCACHE_H 1
3
4   #include <linux/slab.h>
5   #include <linux/slab_def.h>
6   #include <linux/atomic.h>
7   #include <linux/mm.h>
8   #include <linux/fs.h>
9   #include <linux/spinlock.h>
10  #include <linux/bug.h>
11  #include <linux/rcupdate.h>
12
13  #include <rb_voidp.h>
14
15  /* patched table; cached */
16  struct tcache_table {
17          union {
18                  struct file_operations          f_op;
19                  struct vm_operations_struct     vm_ops;
20                  struct address_space_operations a_ops;
21          };
22          atomic_t ref_cnt;
23          struct rcu_head rcu;
24
25          RB_VOIDP_EMBED(void *unpatched);
26  };
27  RB_VOIDP_TYPED(__tcache_table, struct tcache_table)
28
29  extern atomic_t __rb_tcache_cnt;
30  extern struct kmem_cache *__tcache_table;
31
32  static inline int tcache_init(void) {
33          return (__tcache_table = KMEM_CACHE(tcache_table, SLAB_HWCACHE_ALIGN))
34            != NULL;
35  }
36
37  static inline void tcache_cleanup(void) {
38          kmem_cache_destroy(__tcache_table);
39  }
40
41  /**
42   * tcache_ispatched - return non-0 if @tab is patched
43   * @tab: pointer to table
44   */
45  static inline int tcache_ispatched(void *tab) {
46          return virt_to_head_page(tab)->slab_cache == __tcache_table;
47  }
48
49  typedef void (*__patch_table_t)(void *unpatched, struct tcache_table *patched);
50  void *tcache_get(void *, __patch_table_t);
51  void *tcache_put(void *);
52
53  /**
54   * tcache_assign - assign @_t table
55   * @_p: pointer to object operations table
56   * @_t: operations table
57   */
58  #define tcache_assign(_p, _t) rcu_assign_pointer(_p, _t)
```

```
59
60  /**
61   * tcache_getunpatched - get unpatched table for @_t
62   * @_t: operations table
63   */
64  #define tcache_getunpatched(_t)                                              \
65    ({                                                                         \
66      struct tcache_table *____t = (struct tcache_table *)                     \
67                            ({ rcu_read_lock(); rcu_dereference(_t); });       \
68      void *____r = tcache_ispatched(____t) ? ____t->unpatched : ____t;        \
69      rcu_read_unlock(); ____r;                                                \
70    })
71
72  #endif
```

D.5. src/src-klinux/tcache.c

```
1   #include <tcache.h>
2
3   DEFINE_SPINLOCK(__rb_tcache_lock); /* protects __rb_tcache */
4   struct rb_root __rb_tcache = RB_ROOT;
5   atomic_t __rb_tcache_cnt = ATOMIC_INIT(0);
6
7   struct kmem_cache *__tcache_table;
8
9   /**
10   * tcache_get - get a reference to patched table
11   * @unpatched: pointer to unpatched table
12   * @patch_fn: function to get a patched table
13   *
14   * Get patched table of @unpatched either from the cache or calling patch_fn;
15   * calling tcache_get() on a patched table increments reference counter and
16   * returns it.
17   */
18  void *tcache_get(void *unpatched, __patch_table_t patch_fn) {
19          struct tcache_table *t = (struct tcache_table *)unpatched;
20          struct rb_node **p, *parent;
21
22          if (tcache_ispatched(t))
23                  goto inc_refcnt; /* request to patch a patched table */
24
25          spin_lock(&__rb_tcache_lock);
26          t = __tcache_table_lookup_hint(&__rb_tcache, unpatched, &parent, &p);
27          if (unlikely(!t)) {
28                  t = kmem_cache_alloc(__tcache_table, GFP_ATOMIC);
29                  if (!t)
30                          return spin_unlock(&__rb_tcache_lock), NULL;
31
32                  patch_fn(unpatched, t);
33                  __rb_voidp_add_hint(&__rb_tcache, &t->__rb_voidp_ent, parent, p);
34                  atomic_inc(&__rb_tcache_cnt);
35          }
36          spin_unlock(&__rb_tcache_lock);
37
38  inc_refcnt:
39          atomic_inc(&t->ref_cnt);
40          return t;
41  }
42
43  static void __tcache_free_rcu(struct rcu_head *rcu) {
44          kmem_cache_free(__tcache_table,
45                          container_of(rcu, struct tcache_table, rcu));
```

```
46  }
47
48  /**
49   * tcache_put - drop a reference to patched table
50   * @patched: pointer to patched table
51   *
52   * Unreference @patched and return its unpatched version; patched tables
53   * will be freed if ref_cnt reaches 0. Calling tcache_put() on an unpatched
54   * table is undefined.
55   */
56  void *tcache_put(void *patched) {
57          struct tcache_table *t = (struct tcache_table *)patched;
58          void *unpatched = t->unpatched;
59
60          if (atomic_dec_and_test(&t->ref_cnt)) {
61                  spin_lock(&__rb_tcache_lock);
62                  rb_erase(&t->__rb, &__rb_tcache);
63                  spin_unlock(&__rb_tcache_lock);
64                  atomic_dec(&__rb_tcache_cnt);
65
66                  call_rcu(&t->rcu, __tcache_free_rcu);
67          }
68          return unpatched;
69  }
```

D.6. include/src-klinux/mcast.h

```
1   #ifndef _MCAST_H
2   #define _MCAST_H 1
3
4   #include <linux/slab.h>
5   #include <linux/slab_def.h>
6   #include <linux/atomic.h>
7   #include <linux/mm.h>
8   #include <linux/rculist.h>
9   #include <linux/rwsem.h>
10
11  #include <rb_voidp.h>
12
13  struct mcast_group;
14  struct mcast_subscriber_external;
15
16  struct mcast_group_operations {
17          void (*no_subscribers)(struct mcast_group *);
18          void (*no_publishers)(struct mcast_group *);
19  };
20
21  struct mcast_group {
22          struct rb_root subscribers;
23          struct rw_semaphore s_rwsem;     /* protect subscribers; changed from rwlock_t
24                                            * to allow sleeping */
25
26          atomic_t publishers;
27          int shared : 1;
28  #define MCAST_GROUP_PRIVATE 0
29  #define MCAST_GROUP_SHARED  1
30          struct mcast_group_operations *ops;
31
32          struct rcu_head rcu;
33          struct list_head list; /* link in __mcast_list */
34  };
35
```

```
36  struct mcast_subscriber {
37          RB_VOIDP_EMBED(struct mcast_subscriber_external *sp);
38
39          struct list_head list; /* link in initial subscriptor list
40                                  * for mcast_create_group(); unused after this */
41  };
42  RB_VOIDP_TYPED(__mcast_subscriber, struct mcast_subscriber)
43
44  /**
45   * mcast_for_each_subscriber - iterate over mcast_group
46   * @_s: struct mcast_subscriber *
47   * @_g: struct mcast_group *
48   *
49   * Iterate over mcast_group with @_g->s_rwsem down
50   */
51  #define mcast_for_each_subscriber(_s, _g)                                          \
52    for (_s = rb_entry_safe(rb_first(&(_g)->subscribers), struct mcast_subscriber, __rb); \
53         _s; _s = rb_entry_safe(rb_next(&(_s)->__rb),                              \
54                                struct mcast_subscriber, __rb))
55
56  extern struct list_head __mcast_list;
57  extern atomic_t __mcast_list_cnt;
58
59  extern struct kmem_cache *__mcast_group;
60  extern struct kmem_cache *__mcast_subscriber;
61
62  static inline void mcast_init(void) {
63          __mcast_group = KMEM_CACHE(mcast_group, SLAB_HWCACHE_ALIGN);
64          __mcast_subscriber = KMEM_CACHE(mcast_subscriber, SLAB_HWCACHE_ALIGN);
65  }
66
67  static inline void mcast_cleanup(void) {
68          kmem_cache_destroy(__mcast_subscriber);
69          kmem_cache_destroy(__mcast_group);
70  }
71
72  struct mcast_group *mcast_create_group(struct list_head *initial, int shared,
73                                         struct mcast_group_operations *);
74  void mcast_delete_group(struct mcast_group *grp);
75
76  /**
77   * mcast_subscriber_new - return new struct mcast_subscriber
78   * @sp: the subscriber
79   */
80  static inline struct mcast_subscriber *mcast_subscriber_new(struct
        mcast_subscriber_external *sp) {
81          struct mcast_subscriber *s = kmem_cache_alloc(__mcast_subscriber,
82                                                        GFP_KERNEL);
83          if (s)
84                  s->sp = sp;
85          return s;
86  }
87
88  struct mcast_group *
89  mcast_merge_group(struct mcast_group *_a, struct mcast_group *_b, int shared);
90
91  /**
92   * mcast_publisher_join - publisher joins a group
93   * @grp: group to join
94   */
95  static inline void mcast_publisher_join(struct mcast_group *grp) {
96          atomic_inc(&grp->publishers);
97  }
98
99  /**
100  * mcast_publisher_leave - publisher leaves a group
101  * @grp: group to leave
```

```
102  */
103 static inline void mcast_publisher_leave(struct mcast_group *grp) {
104         if (atomic_dec_and_test(&grp->publishers) && grp->ops->no_publishers)
105                 grp->ops->no_publishers(grp);
106 }
107
108 /**
109  * mcast_subscriber_leave - subscriber leaves a group
110  * @grp: group to leave
111  * @sp: the subscriber
112  */
113 static inline void mcast_subscriber_leave(struct mcast_group *grp, struct
        mcast_subscriber_external *sp) {
114         down_write(&grp->s_rwsem);
115         __mcast_subscriber_delete(&grp->subscribers, sp);
116         up_write(&grp->s_rwsem);
117         if (RB_EMPTY_ROOT(&grp->subscribers) && grp->ops->no_subscribers)
118                 grp->ops->no_subscribers(grp);
119 }
120
121 /**
122  * mcast_subscriber_quit - subscriber quits
123  * @sp: the subscriber
124  */
125 static inline void mcast_subscriber_quit(struct mcast_subscriber_external *sp) {
126         struct mcast_group *grp;
127
128         rcu_read_lock();
129         list_for_each_entry_rcu(grp, &__mcast_list, list)
130                 mcast_subscriber_leave(grp, sp);
131         rcu_read_unlock();
132 }
133
134 #endif
```

D.7. src/src-klinux/mcast.c

```
1  #include <mcast.h>
2
3  #include <linux/bug.h>
4  #include <linux/spinlock.h>
5
6  DEFINE_SPINLOCK(__mcast_list_lock); /* protect __mcast_list */
7  LIST_HEAD(__mcast_list);
8  atomic_t __mcast_list_cnt = ATOMIC_INIT(0);
9
10 struct kmem_cache *__mcast_group;
11 struct kmem_cache *__mcast_subscriber;
12
13 /**
14  * __mcast_group_new - alloc and return new struct mcast_group
15  * @shared: shared groups will copy-on-write if modified.
16  * @ops: struct mcast_group_operations
17  */
18 static inline struct mcast_group *__mcast_group_new(int shared, struct
        mcast_group_operations *ops) {
19         struct mcast_group *r = kmem_cache_alloc(__mcast_group, GFP_KERNEL);
20
21         if (r)
22                 *r = (struct mcast_group) { RB_ROOT, __RWSEM_INITIALIZER(r->s_rwsem),
                        ATOMIC_INIT(0), shared, ops };
23         return r;
```

```
24  }
25
26  static inline void __mcast_register_group(struct mcast_group *grp) {
27          spin_lock(&__mcast_list_lock);
28          list_add_rcu(&grp->list, &__mcast_list);
29          spin_unlock(&__mcast_list_lock);
30
31          atomic_inc(&__mcast_list_cnt);
32  }
33
34  static inline void __mcast_unregister_group(struct mcast_group *grp) {
35          spin_lock(&__mcast_list_lock);
36          list_del_rcu(&grp->list);
37          spin_unlock(&__mcast_list_lock);
38
39          atomic_dec(&__mcast_list_cnt);
40  }
41
42  /**
43   * mcast_create_group - create a multicast group
44   * @initial: the initial member list
45   * @shared: specifies if the group is shared
46   * @ops: struct mcast_group_operations
47   *
48   * Create multicast group with initial subscriber list; the list member of
49   * struct mcast_subscriber is unused after this.
50   */
51  struct mcast_group *mcast_create_group(struct list_head *initial, int shared,
52                                         struct mcast_group_operations *ops) {
53          struct mcast_group *grp = __mcast_group_new(shared, ops);
54          struct mcast_subscriber *subscriber;
55
56          if (!grp)
57                  goto nomem;
58
59          down_write(&grp->s_rwsem);
60          list_for_each_entry(subscriber, initial, list) {
61                  __mcast_subscriber_add(&grp->subscribers, subscriber);
62          }
63          up_write(&grp->s_rwsem);
64
65          __mcast_register_group(grp);
66  nomem:
67          return grp;
68  }
69
70  void __mcast_free_group_rcu(struct rcu_head *rcu) {
71          struct mcast_group *grp = container_of(rcu, struct mcast_group, rcu);
72          struct mcast_subscriber *subscriber, *p;
73
74          down_write(&grp->s_rwsem);
75          rbtree_postorder_for_each_entry_safe(subscriber, p, &grp->subscribers,
76   __rb) {
77                  kmem_cache_free(__mcast_subscriber, subscriber);
78          }
79          up_write(&grp->s_rwsem);
80          kmem_cache_free(__mcast_group, grp);
81  }
82
83  void mcast_delete_group(struct mcast_group *grp) {
84          BUG_ON(atomic_read(&grp->publishers) != 0);
85          __mcast_unregister_group(grp);
86
87          /* see Documentation/RCU/UP.txt, example 2 */
88          call_rcu(&grp->rcu, __mcast_free_group_rcu);
89  }
90
```

```
91  void __mcast_merge_onto(struct mcast_group *dest, struct mcast_group *src) {
92          struct rb_node **p, *parent, *i;
93          struct mcast_subscriber *s;
94
95          down_write(&dest->s_rwsem);
96          down_read(&src->s_rwsem);
97
98          for (i = rb_first(&src->subscribers); i != NULL; i = rb_next(i)) {
99                  s = rb_entry(i, struct mcast_subscriber, __rb);
100
101                 /* use "__rb_voidp__xxx" functions to avoid allocation of a new
102                  * struct mcast_subscriber that already is in dest
103                  */
104                 if (!__rb_voidp_lookup(&dest->subscribers, s->sp, &parent, &p))
105                         __rb_voidp_add_hint(&dest->subscribers,
106                                             &(mcast_subscriber_new(s->sp)->__rb_voidp_ent
                                                ), parent, p);
107         }
108
109         up_read(&src->s_rwsem);
110         up_write(&dest->s_rwsem);
111 }
112
113 /**
114  * mcast_merge_group - merge two multicast groups
115  * @_a: merge onto this group
116  * @_b: group to merge
117  * @shared: specifies if the new group is shared if a copy of _a was used
118  *
119  * Merge group _b onto group _a and return result; if _a is shared, a copy
120  * of it is made before merging.
121  */
122 struct mcast_group *
123 mcast_merge_group(struct mcast_group *_a, struct mcast_group *_b, int shared) {
124         struct mcast_group *ret = _a;
125
126         if (ret->shared) {
127                 ret = __mcast_group_new(shared, ret->ops);
128                 if (!ret)
129                         goto nomem;
130                 __mcast_register_group(ret);
131                 __mcast_merge_onto(ret, _a);
132         }
133         __mcast_merge_onto(ret, _b);
134
135 nomem:
136         return ret;
137 }
```

D.8. include/src-klinux/koh_mgmt.h

```
1  #ifndef _KOH_MGMT_H
2  #define _KOH_MGMT_H 1
3
4  #include <linux/slab.h>
5  #include <linux/slab_def.h>
6  #include <linux/atomic.h>
7  #include <linux/spinlock.h>
8
9  #include <tcache.h>
10 #include <mcast.h>
11
```

```
12  struct koh_object {
13          void **object_ops; /* pointer to ops table within object */
14          atomic_t count;    /* hook usage */
15
16          RB_VOIDP_EMBED(void *object);
17  };
18  RB_VOIDP_TYPED(__koh_object, struct koh_object)
19
20  struct koh_delivery {
21          struct mcast_group *recipient;
22          struct koh_object *koh_obj;
23
24          struct list_head list;          /* link in __koh_list_delivery for traversal in
                koh_abort()
25                                           * as rbtree_postorder_for_each_entry_safe does
                                                 not work */
26
27          RB_VOIDP_EMBED(void *publisher); /* any kernel object */
28  };
29  RB_VOIDP_TYPED(__koh_delivery, struct koh_delivery)
30
31  extern rwlock_t __koh_rb_lock;
32  extern struct rb_root __koh_rb_object;
33  extern struct rb_root __koh_rb_delivery;
34  extern atomic_t __koh_rb_object_cnt;
35
36  extern struct kmem_cache *__koh_object;
37  extern struct kmem_cache *__koh_delivery;
38
39  static inline void koh_init(void) {
40          __koh_object = KMEM_CACHE(koh_object, SLAB_HWCACHE_ALIGN);
41          __koh_delivery = KMEM_CACHE(koh_delivery, SLAB_HWCACHE_ALIGN);
42  }
43
44  static inline void koh_cleanup(void) {
45          kmem_cache_destroy(__koh_delivery);
46          kmem_cache_destroy(__koh_object);
47  }
48
49  int __koh_hook(void *object, void *publisher, void **object_ops, __patch_table_t,
50                 struct mcast_group *recipient);
51  /* This macro avoids compiler warning on object_ops type */
52  #define koh_hook(object, publisher, object_ops, patch_fn, recipient) \
53    __koh_hook(object, publisher, (void **)object_ops, patch_fn, recipient)
54
55  void __koh_unhook_unlocked(struct koh_delivery *d);
56  void koh_abort(struct mcast_group *grp);
57
58  /**
59   * koh_unhook - stop @publisher and unhook object if no publishers need it
60   * @publisher: publisher object
61   *
62   * Stop publishing for @publisher and unhook its koh_object if there
63   * are no more publishers.
64   */
65  static inline int koh_unhook(void *publisher) {
66          return ({ struct koh_delivery *____d; write_lock(&__koh_rb_lock); ____d =
                __koh_delivery_lookup(&__koh_rb_delivery, publisher); if (likely(____d))
                __koh_unhook_unlocked(____d); write_unlock(&__koh_rb_lock); ____d != NULL; })
                ;
67  }
68
69  /**
70   * koh_getrecipient - get delivery group for @publisher
71   * @publisher: publisher object
72   *
73   * Get delivery group for @publisher and return with the group semaphore down
```

```
74   */
75  static inline struct mcast_group *koh_getrecipient(void *publisher) {
76          struct koh_delivery *____d;
77
78          read_lock(&__koh_rb_lock);
79          ____d = __koh_delivery_lookup(&__koh_rb_delivery, publisher);
80          read_unlock(&__koh_rb_lock);
81          /* group is warranted not to be freed while s_rwsem is down; see
                __mcast_free_group_rcu */
82          return ____d ? ({ down_read(&____d->recipient->s_rwsem); ____d->recipient; }) :
                NULL;
83  }
84
85  /**
86   * koh_putrecipient - put a reference to 'struct mcast_group *'
87   * @grp: pointer to 'struct mcast_group'
88   *
89   * Put a reference to 'struct mcast_group *' previously get with koh_getrecipient()
90   */
91  #define koh_putrecipient(grp) up_read(&(grp)->s_rwsem)
92
93  #endif
```

D.9. src/src-klinux/koh_mgmt.c

```
1   #include <linux/errno.h>
2   #include <linux/printk.h>
3
4   #include <koh_mgmt.h>
5
6   DEFINE_RWLOCK(__koh_rb_lock); /* protect "__koh_rb_xxx" rb_root, __koh_list_delivery */
7   struct rb_root __koh_rb_object = RB_ROOT;
8   struct rb_root __koh_rb_delivery = RB_ROOT;
9   LIST_HEAD(__koh_list_delivery);
10  atomic_t __koh_rb_object_cnt = ATOMIC_INIT(0);
11
12  struct kmem_cache *__koh_object;
13  struct kmem_cache *__koh_delivery;
14
15  /**
16   * __koh_hook - hook @object on behalf of @publisher
17   * @object: kernel object to hook
18   * @publisher: publisher object
19   * @object_ops: pointer to @object operations table
20   * @patch_fn: function to patch @object_ops
21   * @recipient: mcast_group that will receive messages
22   *
23   * Hook @object on behalf of @publisher and deliver messages to @recipient; @object_ops
24   * table will be patched using @patch_fn. If @recipient is already registered,
25   * it will be merge onto the current delivery group.
26   *
27   * Return 0 or -ENOMEM.
28   */
29  int __koh_hook(void *object, void *publisher, void **object_ops, __patch_table_t patch_fn
        ,
30                 struct mcast_group *recipient) {
31          struct rb_node **p, *parent;
32          struct koh_delivery *d;
33
34          write_lock(&__koh_rb_lock);
35          d = __koh_delivery_lookup_hint(&__koh_rb_delivery, publisher, &parent, &p);
36
```

```
37          if (!d) { /* unregistered publisher */
38                  d = kmem_cache_alloc(__koh_delivery, GFP_ATOMIC);
39                  if (!d)
40                          { write_unlock(&__koh_rb_lock); return -ENOMEM; }
41
42                  *d = (struct koh_delivery) { .recipient = recipient, .publisher =
                        publisher };
43                  mcast_publisher_join(recipient);
44                  __rb_voidp_add_hint(&__koh_rb_delivery, &d->__rb_voidp_ent, parent, p);
45                  list_add(&d->list, &__koh_list_delivery);
46
47                  d->koh_obj = __koh_object_lookup_hint(&__koh_rb_object, object, &parent,
                        &p);
48                  if (!d->koh_obj) { /* object was in unhooked state */
49                          d->koh_obj = kmem_cache_alloc(__koh_object, GFP_ATOMIC);
50                          if (!d->koh_obj)
51                                  { write_unlock(&__koh_rb_lock); return -ENOMEM; }
52
53                          atomic_inc(&__koh_rb_object_cnt);
54                          *d->koh_obj = (struct koh_object) { .object_ops = object_ops,
55                                                  .count = ATOMIC_INIT(0), .object = object
                                                      };
56                          __rb_voidp_add_hint(&__koh_rb_object, &d->koh_obj->__rb_voidp_ent
                                , parent, p);
57
58                          tcache_assign(*object_ops, tcache_get(*object_ops, patch_fn)); /*
                                patch ops */
59                  }
60                  atomic_inc(&d->koh_obj->count);
61          } else {
62                  struct mcast_group *merged = mcast_merge_group(d->recipient,
63                                                                 recipient,
                                                                       MCAST_GROUP_PRIVATE);
64
65                  if (merged != d->recipient) { /* d->recipient was shared */
66                          mcast_publisher_leave(d->recipient);
67                          mcast_publisher_join(merged), d->recipient = merged;
68                  }
69          }
70
71          write_unlock(&__koh_rb_lock);
72          return 0;
73  }
74
75  void __koh_unhook_unlocked(struct koh_delivery *d) {
76          struct koh_object *o = d->koh_obj;
77
78          /* unregister @publisher */
79          list_del(&d->list);
80          rb_erase(&d->__rb, &__koh_rb_delivery);
81
82          mcast_publisher_leave(d->recipient);
83          kmem_cache_free(__koh_delivery, d);
84          /* unhook object if no publishers left */
85          if (o && atomic_dec_and_test(&o->count)) {
86                  if (tcache_ispatched(*o->object_ops)) {
87                          tcache_assign(*o->object_ops, tcache_put(*o->object_ops)); /*
                                restore */
88                  } else {
89                          pr_warning("[kmodule] someone else restored %p object ops\n", o->
                                object);
90                  }
91
92                  rb_erase(&o->__rb, &__koh_rb_object);
93                  kmem_cache_free(__koh_object, o);
94                  atomic_dec(&__koh_rb_object_cnt);
95          }
```

```
96  }
97
98  /**
99   * koh_abort - abort delivery to group @grp
100  * @grp: group to which delivery is aborted
101  *
102  * Abort delivery to group @grp, that is call __koh_unhook_unlocked() for each
103  * koh_delivery struct that delivers messages to @grp.
104  * May call no_publishers() on @grp and unhook object if no publishers
105  * remain after this.
106  */
107 void koh_abort(struct mcast_group *grp) {
108         struct koh_delivery *d, *n;
109
110         write_lock(&__koh_rb_lock);
111         list_for_each_entry_safe(d, n, &__koh_list_delivery, list) {
112                 if (d->recipient == grp)
113                         __koh_unhook_unlocked(d);
114         }
115         write_unlock(&__koh_rb_lock);
116 }
```

D.10. include/src-klinux/uapi_ioctl.h

```
1   #ifndef _UAPI_IOCTL_H
2   #define _UAPI_IOCTL_H 1
3
4   #include <linux/types.h>
5   #include <linux/sched.h>
6   #include <linux/ioctl.h>
7
8   #ifndef __KERNEL__
9   /* not defined in uapi/linux/sched.h */
10  #define TASK_COMM_LEN 16
11  #endif
12
13  struct trace_request {
14          int flags;
15  #define TR_ENABLE          0x01
16  #define TR_COMM            0x02
17  #define TR_TGID            0x04
18  #define TR_ACCEPT_DELIVERY 0x08
19  #define TR_WITH_BUFFER     0x10
20          int circbuf_size;                  /* requested size for circular buffer */
21          int trunc_size;                    /* truncate payloads to this value */
22
23          union {
24                  char  comm[TASK_COMM_LEN]; /* used if TR_ENABLE is set */
25                  pid_t tgid;                /* ditto for TR_TGID */
26          };
27  };
28
29  struct trace_stat {
30          int count;
31          int overrun_count; /* not delivered due to buffer overrun */
32  };
33
34  #define IOC_TRACEREQUEST       _IOW('/', 0, struct trace_request)
35  #define IOC_TR_ENABLE          _IOW('/', 1, int)
36  #define IOC_TR_ACCEPT_DELIVERY _IOW('/', 2, int)
37  #define IOC_TRACESTAT          _IOR('/', 3, struct trace_stat)
38
```

```
39 #endif
```

D.11. include/src-klinux/param.h

```
1  #ifndef _PARAM_H
2  #define _PARAM_H 1
3
4  struct kmodule_param {
5          char *sys_call_table;    /* manually specify the sys_call_table symbol
6                                    * address */
7
8          int allow_max;           /* limit concurrent module users */
9          int trace_nonregular;    /* set to 1 to trace non-regular files */
10         int bufsize_min;         /* circular buffer size range */
11         int bufsize_max;
12
13         int null_writepages;     /* set to 1 to nullify writepages method of
14                                   * address_space_operations */
15 };
16 extern struct kmodule_param __param;
17
18 #endif
```

D.12. include/src-klinux/chrdev.h

```
1  #ifndef _CHRDEV_H
2  #define _CHRDEV_H 1
3
4  #include <linux/list.h>
5  #include <linux/circ_buf.h>
6  #include <linux/sched.h>
7  #include <linux/wait.h>
8  #include <linux/semaphore.h>
9  #include <linux/spinlock.h>
10 #include <linux/slab.h>
11 #include <linux/slab_def.h>
12 #include <linux/atomic.h>
13 #include <linux/fs.h>
14
15 #include <uapi_ioctl.h>
16
17 struct mcast_subscriber_external {
18         struct circ_buf   buf;
19         int               buf_size; /* size of circular buffer */
20         struct semaphore  sem;      /* protect circ_buf writers */
21         wait_queue_head_t wq;       /* wait queue for buf, sleep using
               wait_event_interruptible_exclusive() */
22
23         __u32 seq;                  /* sequence number of next message */
24
25         struct trace_stat stat;
26 };
27
28 struct trace_info {
29         struct list_head     list;    /* link in __trace_info_list */
30         struct task_struct   *tracer;
31         struct trace_request req;
32         spinlock_t           req_lock; /* protect req */
```

```
33
34          struct mcast_subscriber_external subscriber; /* can subscribe to mcast_group */
35  };
36
37  extern rwlock_t __trace_info_lock;
38  extern struct list_head __trace_info_list;
39  extern atomic_t __trace_info_cnt;
40
41  extern struct kmem_cache *__trace_info;
42
43  int thischrdev_init(void);
44  void thischrdev_cleanup(void);
45  void thischrdev_getsubscribers_lock(struct file *filp, const char *filename,
46                                      struct list_head *subscribers);
47
48  /**
49   * thischrdev_getsubscribers_unlock - unlock __trace_info_lock after a call to
         thischrdev_getsubscribers_lock()
50   *
51   * See thischrdev_getsubscribers_lock() documentation for an explanation.
52   */
53  #define thischrdev_getsubscribers_unlock() read_unlock(&__trace_info_lock)
54
55  #endif
```

D.13. src/src-klinux/chrdev.c

```
1   #include <linux/fs.h>
2   #include <linux/module.h>
3   #include <linux/device.h>
4   #include <linux/err.h>
5   #include <linux/bug.h>
6   #include <linux/vmalloc.h>
7   #include <linux/log2.h>
8   #include <asm/uaccess.h>
9
10  #include <chrdev.h>
11  #include <mcast.h>
12  #include <param.h>
13
14  DEFINE_RWLOCK(__trace_info_lock); /* protect __trace_info_list */
15  LIST_HEAD(__trace_info_list);
16  atomic_t __trace_info_cnt = ATOMIC_INIT(0);
17
18  struct kmem_cache *__trace_info;
19
20  /**
21   * __traceinfo_create - initialize and register new 'struct trace_info'
22   *
23   * Initialize and register a new 'struct trace_info'; a pointer to the
24   * allocated struct is returned.
25   */
26  static struct trace_info *__traceinfo_create(void) {
27          struct trace_info *ret = kmem_cache_alloc(__trace_info, GFP_KERNEL);
28
29          if (!ret)
30                  return NULL;
31
32          *ret = (struct trace_info) { .tracer = current, .req_lock = __SPIN_LOCK_UNLOCKED(
                ret->req_lock), .subscriber.sem = __SEMAPHORE_INITIALIZER(ret->subscriber.sem
                , 1), .subscriber.wq = __WAIT_QUEUE_HEAD_INITIALIZER(ret->subscriber.wq), .
                subscriber.seq = 0 };
```

```
33
34          /* register new 'struct trace_info' */
35          write_lock(&__trace_info_lock);
36          list_add_tail(&ret->list, &__trace_info_list);
37          write_unlock(&__trace_info_lock);
38
39          return ret;
40  }
41
42  /**
43   * __traceinfo_destroy - unregister and destroy a 'struct trace_info'
44   * @ti: 'struct trace_info *' to destroy
45   *
46   * Unregister and destroy a 'struct trace_info' previously registed
47   * calling __traceinfo_create()
48   */
49  static void __traceinfo_destroy(struct trace_info *this) {
50          /* abort current readers */
51          this->req.flags &= ~(TR_ENABLE | TR_ACCEPT_DELIVERY);
52          wake_up_interruptible_all(&this->subscriber.wq);
53
54          /* unregister 'struct trace_info' */
55          write_lock(&__trace_info_lock);
56          list_del(&this->list);
57          write_unlock(&__trace_info_lock);
58
59          mcast_subscriber_quit(&this->subscriber);
60
61          if (this->subscriber.buf.buf)
62                  vfree(this->subscriber.buf.buf);
63          kmem_cache_free(__trace_info, this);
64  }
65
66  static inline int __must_subscribe(struct trace_info *this, struct file *filp,
67                                     const char *filename) {
68          return ({ int ____ret; spin_lock(&this->req_lock);
69                    ____ret = (this->req.flags & TR_ENABLE)
70                              /* check task uid */
71                              && (uid_eq(task_uid(this->tracer), current_uid()) ||
                                    __kuid_val(task_uid(this->tracer)) == 0)
72                              /* check comm/tgid */
73                              && (((this->req.flags & TR_COMM) && strcmp(current->comm,
                                    this->req.comm) == 0)
74                                  || ((this->req.flags & TR_TGID) && current->tgid == this
                                        ->req.tgid)
75                                  || !(this->req.flags & (TR_COMM | TR_TGID)))
76                              /* check S_ISREG */
77                              && (filp ? S_ISREG(filp->f_inode->i_mode) || __param.
                                    trace_nonregular : 1);
78                    spin_unlock(&this->req_lock); ____ret; });
79  }
80
81  /**
82   * thischrdev_getsubscribers_lock - get list of subscribers for @filp and @path
83   * @filp: struct file *
84   * @filename: path used to open @filp
85   * @subscribers: returned list
86   *
87   * Get list of struct mcast_subscribers that may trace @filp. Return with
88   * __trace_info_lock locked; caller should call thischrdev_getsubscribers_unlock().
89   *
90   * This ensures __traceinfo_destroy() does not call mcast_subscriber_quit() before
91   * the caller creates mcast_group.
92   */
93  void thischrdev_getsubscribers_lock(struct file *filp, const char *filename,
94                                      struct list_head *subscribers) {
95          struct trace_info *iter;
```

```
96
97          read_lock(&__trace_info_lock);
98          list_for_each_entry(iter, &__trace_info_list, list) {
99                  if (__must_subscribe(iter, filp, filename))
100                         list_add_tail(&mcast_subscriber_new(&iter->subscriber)->list,
101                                       subscribers);
102         }
103 }
104
105 /**
106  * __circ_buf_read - copy from circular buffer to userspace
107  * @s: 'struct mcast_subscriber_external *' to copy from
108  * @buf: userspace buffer
109  * @count: byte count
110  *
111  * Copy from @s circular buffer to @buf buffer in userspace
112  */
113 static ssize_t __circ_buf_read(struct mcast_subscriber_external *s, char __user *buf,
114                                int count) {
115         ssize_t ret = 0;
116
117         while (count > 0) {
118                 int n = min(count, CIRC_CNT_TO_END(s->buf.head, s->buf.tail,
119                                                    s->buf_size));
120                 if (copy_to_user(buf, &s->buf.buf[s->buf.tail], n))
121                         break;
122
123                 s->buf.tail = (s->buf.tail + n) & (s->buf_size - 1);
124                 count -= n;
125                 buf   += n;
126                 ret   += n;
127         }
128         return ret;
129 }
130
131 static ssize_t __chrdev_read(struct file *filp, char __user *buf, size_t count,
132                              loff_t *off) {
133         struct trace_info *this = (struct trace_info *)filp->private_data;
134
135         if (filp->f_flags & O_NONBLOCK)
136                 return -EAGAIN;
137         if (wait_event_interruptible_exclusive(this->subscriber.wq,
138                                                (this->subscriber.buf.head != this->
                                                    subscriber.buf.tail)
139                                                || !(this->req.flags & TR_ENABLE)))
140                 return -ERESTARTSYS;
141
142         /* return -EBUSY if the trace was aborted */
143         if (!(this->req.flags & TR_ENABLE))
144                 return -EBUSY;
145
146         return __circ_buf_read(&this->subscriber, buf, count);
147 }
148
149 static long __chrdev_unlocked_ioctl(struct file *filp, unsigned int cmd,
150                                     unsigned long arg) {
151         struct trace_info *this = (struct trace_info *)filp->private_data;
152         struct trace_request tmp;
153
154         if (_IOC_TYPE(cmd) != '/')
155                 return -ENOTTY;
156         if ((_IOC_DIR(cmd) & _IOC_READ) && !access_ok(VERIFY_WRITE, arg, _IOC_SIZE(cmd)))
157                 return -EFAULT;
158         if ((_IOC_DIR(cmd) & _IOC_WRITE) && !access_ok(VERIFY_READ, arg, _IOC_SIZE(cmd)))
159                 return -EFAULT;
160
161         switch (cmd) {
```

```
162         case IOC_TRACEREQUEST:
163                 if ((this->req.flags & TR_ENABLE) || copy_from_user(&tmp, (void __user *)
                        arg, _IOC_SIZE(cmd)))
164                         return -EFAULT;
165
166                 /* allocate circular buffer */
167                 if (this->subscriber.buf.buf)
168                         vfree(this->subscriber.buf.buf);
169                 this->subscriber.buf.buf = vmalloc( clamp_val(roundup_pow_of_two(tmp.
                        circbuf_size),
170                                                               __param.bufsize_min,
171                                                               __param.bufsize_max) );
172                 if (!this->subscriber.buf.buf)
173                         return -EFAULT;
174
175                 spin_lock(&this->req_lock);
176                 this->req = tmp;
177                 spin_unlock(&this->req_lock);
178                 break;
179
180         case IOC_TR_ENABLE:
181                 if (arg) {
182                         if (this->subscriber.buf.buf) /* cannot enable if buf was not
                                allocated */
183                                 this->req.flags |= TR_ENABLE;
184                 } else {
185                         this->req.flags &= ~TR_ENABLE;
186                         mcast_subscriber_quit(&this->subscriber); /* may unhook kernel
                                objects */
187                 }
188                 break;
189
190         case IOC_TR_ACCEPT_DELIVERY: /* temporary stop receiving messages */
191                 if (arg) {
192                         this->req.flags |= TR_ACCEPT_DELIVERY;
193                 } else {
194                         this->req.flags &= ~TR_ACCEPT_DELIVERY;
195                 }
196                 break;
197
198         case IOC_TRACESTAT:
199                 if (copy_to_user((void __user *)arg, &this->subscriber.stat, _IOC_SIZE(
                        cmd)))
200                         return -EFAULT;
201                 break;
202
203         default:
204                 return -ENOTTY;
205         }
206         return _IOC_SIZE(cmd);
207 }
208
209 static int __chrdev_open(struct inode *inode, struct file *filp) {
210         if (!atomic_add_unless(&__trace_info_cnt, 1, __param.allow_max))
211                 return -EBUSY;    /* if allow_max limit was reached */
212
213         filp->private_data = __traceinfo_create();
214
215         return filp->private_data ? 0 : ({ atomic_dec(&__trace_info_cnt);
216                                            -ENOMEM; });
217 }
218
219 static int __chrdev_release(struct inode *inode, struct file *filp) {
220         __traceinfo_destroy((struct trace_info *)filp->private_data);
221
222         atomic_dec(&__trace_info_cnt);
223         return 0;
```

```
224 }
225
226 static struct file_operations __thischrdev_fop = {
227         .owner = THIS_MODULE,
228         .read = __chrdev_read,
229         .unlocked_ioctl = __chrdev_unlocked_ioctl,
230         .open = __chrdev_open,
231         .release = __chrdev_release,
232 };
233 static int __thischrdev_major;
234 static struct class *__thischrdev_class;
235 static struct device *__thischrdev_device;
236
237 #define THIS_CHRDEV       "kmodule"
238 #define THIS_CHRDEV_CLASS "kmodule-class"
239
240 /**
241  * thischrdev_init - initialize character device
242  *
243  * Initialize and register character device; called from module_init()
244  */
245 int thischrdev_init(void) {
246         int ret = 0;
247
248         /* register chrdev, device class and device */
249         __thischrdev_major = register_chrdev(0, THIS_CHRDEV, &__thischrdev_fop);
250         if (__thischrdev_major < 0)
251                 { ret = __thischrdev_major; goto err_register_chrdev; }
252
253         __thischrdev_class = class_create(THIS_MODULE, THIS_CHRDEV_CLASS);
254         if (IS_ERR(__thischrdev_class))
255                 { ret = PTR_ERR(__thischrdev_class); goto err_class_create; }
256
257         __thischrdev_device = device_create(__thischrdev_class, NULL,
258                                             MKDEV(__thischrdev_major, 0), NULL,
                                                THIS_CHRDEV);
259         if (IS_ERR(__thischrdev_device))
260                 { ret = PTR_ERR(__thischrdev_device); goto err_device_create; }
261
262         /* kmem_cache for struct trace_info */
263         __trace_info = KMEM_CACHE(trace_info, SLAB_HWCACHE_ALIGN);
264         return ret;
265
266 err_device_create:
267         class_destroy(__thischrdev_class);
268 err_class_create:
269         unregister_chrdev(__thischrdev_major, THIS_CHRDEV);
270 err_register_chrdev:
271         return ret;
272 }
273
274 /**
275  * thischrdev_cleanup - cleanup character device before module unload
276  *
277  * Unregister character device; it is a bug to call this if there is any
278  * opened file descriptor.
279  */
280 void thischrdev_cleanup(void) {
281         BUG_ON(atomic_read(&__trace_info_cnt) != 0);
282
283         kmem_cache_destroy(__trace_info);
284
285         device_destroy(__thischrdev_class, MKDEV(__thischrdev_major, 0));
286         class_destroy(__thischrdev_class);
287         unregister_chrdev(__thischrdev_major, THIS_CHRDEV);
288 }
```

D.14. include/src-klinux/sc_hook.h

```
#ifndef _SC_HOOK_H
#define _SC_HOOK_H 1

#include <linux/syscalls.h>
#include <linux/uaccess.h>
#include <linux/errno.h>
#include <asm/unistd.h>
#include <asm/cmpxchg.h>

#ifdef CONFIG_X86
# ifdef CONFIG_X86_32
#  include "sc_hook_x86_32.h"
# else
#  include "sc_hook_x86_64.h"
# endif
#else /* unsupported arch */

/**
 * sc_get_syscalltable - find sys_call_table symbol
 *
 * Find sys_call_table symbol which the kernel does not export; this is
 * architecture-dependent.
 */
static inline unsigned long *sc_get_syscalltable(void) {
        return NULL;
}

/**
 * sc_rw_syscalltable - make sys_call_table page rw
 * @sys_call_table: pointer to sys_call_table
 *
 * Make the sys_call_table page rw as the kernel write-protects it
 */
static inline void sc_rw_syscalltable(unsigned long *sys_call_table) {
}

/**
 * sc_ro_syscalltable - make sys_call_table page ro
 * @sys_call_table: pointer to sys_call_table
 */
static inline void sc_ro_syscalltable(unsigned long *sys_call_table) {
}

#endif

/**
 * sc_is_syscalltable - return non-0 if argument is a valid sys_call_table address
 * @sys_call_table: address of sys_call_table symbol
 */
#define sc_is_syscalltable(sys_call_table) ({ unsigned long ____sys_close; \
  probe_kernel_address(&sys_call_table[__NR_close], ____sys_close) != -EFAULT &&
      ____sys_close == (unsigned long)sys_close; })

struct sc_hook {
        int nr;
        unsigned long *syscall_impl;
};
#define SC_HOOK(__nr, __sy) { __nr, (unsigned long *)__sy }

/**
 * sc_hook_unhook - hook and unhook system calls
 * @sys_call_table: address of the sys_call_table symbol
 * @hookc: size of hookv array
```

```
63   * @hookv: array of 'struct sc_hook'
64   *
65   * Hook and unhook system calls of @hookv; the value of syscall_impl and
66   * the sys_call_table entry are exchanged, so a second call to sc_hook_unhook()
67   * restores the original pointer. sys_call_table is made rw during patching.
68   */
69  static inline void sc_hook_unhook(unsigned long *sys_call_table,
70                                    int hookc, struct sc_hook hookv[]) {
71          int i;
72
73          sc_rw_syscalltable(sys_call_table);
74          for (i = 0; i < hookc; i++) {
75                  *hookv[i].syscall_impl = xchg(&sys_call_table[hookv[i].nr],
76                                                *hookv[i].syscall_impl);
77          }
78          sc_ro_syscalltable(sys_call_table);
79  }
80
81  #endif
```

D.15. include/src-klinux/sc_hook_x86_32.h

```
1   #ifndef _SC_HOOK_X86_32_H
2   #define _SC_HOOK_X86_32_H 1
3
4   #include <asm/desc.h>
5
6   static inline unsigned long *sc_get_syscalltable(void) {
7           struct desc_ptr idt;
8           gate_desc *idt_ent;
9           unsigned char *start, *end;
10
11          native_store_idt(&idt);
12          idt_ent = (gate_desc *)idt.address;
13
14          /* scan through system call handler searching opcode of
15           * 'call *sys_call_table(,%eax,4)' (see arch/x86/kernel/entry_32.S) */
16          for (start = (unsigned char *)gate_offset(idt_ent[0x80]), end = start + 0xff;
17               start < end; ) {
18                  if (*start++ == 0xff && *start++ == 0x14 && *start++ == 0x85)
19                          return *(unsigned long **)start;
20          }
21          return NULL;
22  }
23
24  static inline void sc_rw_syscalltable(unsigned long *sys_call_table) {
25          unsigned int ___level;
26          lookup_address((unsigned long)sys_call_table, &___level)->pte |= _PAGE_RW;
27  }
28
29  static inline void sc_ro_syscalltable(unsigned long *sys_call_table) {
30          unsigned int ___level;
31          lookup_address((unsigned long)sys_call_table, &___level)->pte &= ~_PAGE_RW;
32  }
33
34  #endif
```

D.16. include/src-klinux/sc_hook_x86_64.h

```
#ifndef _SC_HOOK_X86_64_H
#define _SC_HOOK_X86_64_H 1

#include <asm/desc.h>

static inline unsigned long *____scan_syscall_handler(unsigned char *start) {
        unsigned char *end;

        /* scan through system call handler searching opcode of
         * 'call *sys_call_table(,%rax,8)' (see arch/x86/kernel/entry_64.S) */
        for (end = start + 0xff; start < end; ) {
                if (*start++ == 0xff && *start++ == 0x14 && *start++ == 0xc5)
                        return (unsigned long *)(0xffffffff00000000 | (*(unsigned long *)
                            start & 0x00000000ffffffff));
        }
        return NULL;
}

static inline unsigned long *sc_get_syscalltable(void) {
        unsigned long msr_lstar;

        rdmsrl(MSR_LSTAR, msr_lstar); /* LSTAR is system call handler address */
        return ____scan_syscall_handler((unsigned char *)msr_lstar);
}

#ifdef CONFIG_COMPAT
static inline unsigned long *sc_get_syscalltable_ia32(void) {
        struct desc_ptr idt;
        gate_desc *idt_ent;

        native_store_idt(&idt);
        idt_ent = (gate_desc *)idt.address;
        /* interrupt 0x80 is system gate */
        return ____scan_syscall_handler((unsigned char *)idt_ent[0x80]);
}
#endif

static inline void sc_rw_syscalltable(unsigned long *sys_call_table) {
        unsigned int ____level;
        lookup_address((unsigned long)sys_call_table, &____level)->pte |= _PAGE_RW;
}

static inline void sc_ro_syscalltable(unsigned long *sys_call_table) {
        unsigned int ____level;
        lookup_address((unsigned long)sys_call_table, &____level)->pte &= ~_PAGE_RW;
}

#endif
```

D.17. include/src-klinux/hooks.h

```
#ifndef _HOOKS_H
#define _HOOKS_H 1

#include <linux/linkage.h>
#include <linux/list.h>

#include <sc_hook.h>

asmlinkage long ____sys_open(const char __user *filename, int flags, umode_t mode);
asmlinkage long ____sys_openat(int dfd, const char __user *filename, int flags,
```

```
                                umode_t mode);
asmlinkage long ____sys_creat(const char __user *pathname, umode_t mode);

asmlinkage long ____compat_sys_open(const char __user *filename, int flags, umode_t mode)
    ;
asmlinkage long ____compat_sys_openat(int dfd, const char __user *filename, int flags,
                          umode_t mode);

typedef asmlinkage long (*sys_open_t)(const char __user *, int, umode_t);
typedef asmlinkage long (*sys_openat_t)(int, const char __user *, int, umode_t);
typedef asmlinkage long (*sys_creat_t)(const char __user *, umode_t);

extern sys_open_t   __orig_sys_open;
extern sys_openat_t __orig_sys_openat;
extern sys_creat_t  __orig_sys_creat;

struct sc_hook __syscalls[] = { SC_HOOK(__NR_open, &__orig_sys_open),
                                SC_HOOK(__NR_openat, &__orig_sys_openat),
                                SC_HOOK(__NR_creat, &__orig_sys_creat) };

#if defined(CONFIG_X86) && defined(CONFIG_COMPAT)
extern sys_open_t   __orig_compat_sys_open;
extern sys_openat_t __orig_compat_sys_openat;
extern sys_creat_t  __orig_compat_sys_creat;

#include <asm/unistd_x32.h>
struct sc_hook __compat_syscalls[] = { SC_HOOK(__NR_open, &__orig_compat_sys_open),
                                       SC_HOOK(__NR_openat, &__orig_compat_sys_openat),
                                       SC_HOOK(__NR_creat, &__orig_compat_sys_creat) };
#include <asm/unistd.h>

/**
 * hooks_compat_enable_disable - enable/disable syscalls hooks for ia32_sys_call_table
 * @ia32_sys_call_table: ia32_sys_call_table symbol address
 */
static inline void hooks_ia32_enable_disable(unsigned long *ia32_sys_call_table) {
        sc_hook_unhook(ia32_sys_call_table, ARRAY_SIZE(__compat_syscalls),
                       __compat_syscalls);
}
#endif

/**
 * hooks_enable_disable - enable/disable syscall hooks
 * @sys_call_table: sys_call_table symbol address
 */
static inline void hooks_enable_disable(unsigned long *sys_call_table) {
        sc_hook_unhook(sys_call_table, ARRAY_SIZE(__syscalls), __syscalls);
}

#define get_filp(_fd) ({ struct file *____filp;        \
  spin_lock(&current->files->file_lock);               \
  ____filp = files_fdtable(current->files)->fd[_fd];   \
  spin_unlock(&current->files->file_lock); ____filp; })

#endif
```

D.18. src/src-klinux/hooks.c

```
#include <linux/err.h>
#include <linux/fdtable.h>
#include <linux/fs.h>
#include <linux/mm.h>
```

```
5  #include <linux/uaccess.h>
6  #include <linux/gfp.h>
7  #include <linux/module.h>
8  #include <linux/aio.h>
9
10 #include <hooks.h>
11 #include <chrdev.h>
12 #include <koh_mgmt.h>
13 #include <param.h>
14
15 sys_open_t __orig_sys_open = ____sys_open;
16 sys_openat_t __orig_sys_openat = ____sys_openat;
17 sys_creat_t __orig_sys_creat = ____sys_creat;
18
19 #if defined(CONFIG_X86) && defined(CONFIG_COMPAT)
20
21 /* for ia32 compatibility */
22 sys_open_t __orig_compat_sys_open = ____compat_sys_open;
23 sys_openat_t __orig_compat_sys_openat = ____compat_sys_openat;
24 sys_creat_t __orig_compat_sys_creat = ____sys_creat;
25 #endif
26
27 /*
28  *  address space operations
29  */
30 int __aops_writepage(struct page *page, struct writeback_control *wbc) {
31         struct address_space_operations *a_ops = tcache_getunpatched(page->mapping->a_ops
              );
32
33         /* ENTREGA DE MENSAJES OMITIDA PARA FACILITAR LECTURA */
34
35         koh_unhook(page); /* might unhook address_space if we are not waiting
36                            * more pages */
37         return a_ops->writepage(page, wbc);
38 }
39
40 static void __patch_aops(void *unpatched, struct tcache_table *patched) {
41         struct address_space_operations *__unpatched = (struct address_space_operations
              *)unpatched;
42
43         /* copy table */
44         memcpy(&patched->a_ops, __unpatched, sizeof(__unpatched));
45
46         /* fixup */
47         patched->a_ops.writepage = __aops_writepage;
48         if (__param.null_writepages)
49                 patched->a_ops.writepages = NULL;
50
51         atomic_set(&patched->ref_cnt, 0);
52         patched->unpatched = unpatched;
53 }
54
55 /*
56  *  vma operations
57  */
58 static void
59 __patch_vmops(void *, struct tcache_table *);
60
61 void __vmops_open(struct vm_area_struct *area) {
62         struct vm_operations_struct *vm_ops = tcache_getunpatched(area->vm_ops);
63         struct mcast_group *grp = koh_getrecipient(area->vm_file);
64
65         /* this is called from __split_vma(); see mm/mmap.c
66          *
67          * Register new area as publisher if area->vm_file is still a publisher,
68          * else restore unpatched table (__split_vma() made area inherit a patched
69          * table)
```

```
70              */
71              if (grp) {
72                      koh_hook(area, area, &area->vm_ops, __patch_vmops, grp);
73                      koh_putrecipient(grp);
74              } else {
75                      tcache_assign(area->vm_ops, vm_ops);
76              }
77
78              if (vm_ops->open)
79                      vm_ops->open(area);
80      }
81
82      void __vmops_close(struct vm_area_struct *area) {
83              struct vm_operations_struct *vm_ops = tcache_getunpatched(area->vm_ops);
84
85              /* ENTREGA DE MENSAJES OMITIDA PARA FACILITAR LECTURA */
86
87              /* objects should be unhooked before they are reclaimed */
88              koh_unhook(area);
89
90              if (vm_ops->close)
91                      vm_ops->close(area);
92      }
93
94      int __vmops_fault(struct vm_area_struct *vma, struct vm_fault *vmf) {
95              struct vm_operations_struct *vm_ops = tcache_getunpatched(vma->vm_ops);
96
97              /* ENTREGA DE MENSAJES OMITIDA PARA FACILITAR LECTURA */
98
99              return vm_ops->fault(vma, vmf);
100     }
101
102     int __vmops_page_mkwrite(struct vm_area_struct *vma, struct vm_fault *vmf) {
103             struct vm_operations_struct *vm_ops = tcache_getunpatched(vma->vm_ops);
104             int ret = vm_ops->page_mkwrite(vma, vmf);
105             struct address_space *f_mapping = vma->vm_file->f_mapping;
106
107             if (unlikely(ret & (VM_FAULT_ERROR | VM_FAULT_NOPAGE)))
108                     return ret;
109
110             /* only hook address_space objects for shared maps */
111             if ((vma->vm_flags & VM_SHARED) && f_mapping->a_ops->writepage) {
112                     struct mcast_group *grp = koh_getrecipient(vma);
113
114                     if (grp) {
115                             /* ENTREGA DE MENSAJES OMITIDA PARA FACILITAR LECTURA */
116
117                             koh_hook(f_mapping, vmf->page, &f_mapping->a_ops, __patch_aops,
118                                     grp); /* deliver to same recipient as vma */
119
120                             koh_putrecipient(grp);
121                     }
122             }
123             return ret;
124     }
125
126     static void __patch_vmops(void *unpatched, struct tcache_table *patched) {
127             struct vm_operations_struct *__unpatched = (struct vm_operations_struct *)
                unpatched;
128
129             /* copy table */
130             memcpy(&patched->vm_ops, __unpatched, sizeof(__unpatched));
131
132             /* fixup */
133             patched->vm_ops.open = __vmops_open;
134             patched->vm_ops.close = __vmops_close;
135             if (__unpatched->fault)
```

```
136                     patched->vm_ops.fault = __vmops_fault;
137             if (__unpatched->page_mkwrite)
138                     patched->vm_ops.page_mkwrite = __vmops_page_mkwrite;
139
140             atomic_set(&patched->ref_cnt, 0);
141             patched->unpatched = unpatched;
142     }
143
144     /*
145      *  file operations
146      */
147     ssize_t __fop_read(struct file *filp, char __user *buf, size_t count, loff_t *off) {
148             struct file_operations *f_op = tcache_getunpatched(filp->f_op);
149
150             /* ENTREGA DE MENSAJES OMITIDA PARA FACILITAR LECTURA */
151
152             return f_op->read(filp, buf, count, off);
153     }
154
155     ssize_t __fop_write(struct file *filp, const char __user *buf, size_t count, loff_t *off)
            {
156             struct file_operations *f_op = tcache_getunpatched(filp->f_op);
157
158             /* ENTREGA DE MENSAJES OMITIDA PARA FACILITAR LECTURA */
159
160             return f_op->write(filp, buf, count, off);
161     }
162
163     ssize_t __fop_read_iter(struct kiocb *iocb, struct iov_iter *iter) {
164             struct file_operations *f_op = tcache_getunpatched(iocb->ki_filp->f_op);
165
166             /* ENTREGA DE MENSAJES OMITIDA PARA FACILITAR LECTURA */
167
168             return f_op->read_iter(iocb, iter);
169     }
170
171     ssize_t __fop_write_iter(struct kiocb *iocb, struct iov_iter *iter) {
172             struct file_operations *f_op = tcache_getunpatched(iocb->ki_filp->f_op);
173
174             /* ENTREGA DE MENSAJES OMITIDA PARA FACILITAR LECTURA */
175
176             return f_op->write_iter(iocb, iter);
177     }
178
179     long __fop_unlocked_ioctl(struct file *filp, unsigned int cmd, unsigned long arg) {
180             struct file_operations *f_op = tcache_getunpatched(filp->f_op);
181
182             /* ENTREGA DE MENSAJES OMITIDA PARA FACILITAR LECTURA */
183
184             return f_op->unlocked_ioctl(filp, cmd, arg);
185     }
186
187     int __fop_mmap(struct file *filp, struct vm_area_struct *vma) {
188             struct file_operations *f_op = tcache_getunpatched(filp->f_op);
189             struct mcast_group *grp = koh_getrecipient(filp);
190             int ret = f_op->mmap(filp, vma);
191
192             if (ret == 0 && grp /* make sure filp is still a publisher */
193                 && S_ISREG(filp->f_inode->i_mode)) {
194                     /* ENTREGA DE MENSAJES OMITIDA PARA FACILITAR LECTURA */
195
196                     /* messages will be delivered to same recipient as filp */
197
198                     koh_hook(vma, vma, &vma->vm_ops, __patch_vmops, grp);
199             }
200
201             if (grp)
```

```
202                 koh_putrecipient(grp);
203         return ret;
204 }
205
206 int __fop_release(struct inode *inode, struct file *filp) {
207         struct file_operations *f_op = tcache_getunpatched(filp->f_op);
208
209         /* ENTREGA DE MENSAJES OMITIDA PARA FACILITAR LECTURA */
210
211         koh_unhook(filp);
212         return f_op->release ? f_op->release(inode, filp)
213                              : 0;
214 }
215
216 static void __patch_fop(void *unpatched, struct tcache_table *patched) {
217         struct file_operations *__unpatched = (struct file_operations *)unpatched;
218
219         /* copy table */
220         memcpy(&patched->f_op, __unpatched, sizeof(__unpatched));
221
222         /* fixup */
223         patched->f_op.owner = THIS_MODULE;
224         if (__unpatched->read)
225                 patched->f_op.read = __fop_read;
226         if (__unpatched->write)
227                 patched->f_op.write = __fop_write;
228         if (__unpatched->read_iter)
229                 patched->f_op.read_iter = __fop_read_iter;
230         if (__unpatched->write_iter)
231                 patched->f_op.write_iter = __fop_write_iter;
232         if (__unpatched->unlocked_ioctl)
233                 patched->f_op.unlocked_ioctl = __fop_unlocked_ioctl;
234         if (__unpatched->mmap)
235                 patched->f_op.mmap = __fop_mmap;
236         patched->f_op.release = __fop_release;
237
238         /* force default_file_splice_{read,write}() call; see fs/splice.c
239          * These rely on {read,write}_iter() file operations */
240         patched->f_op.splice_write = NULL;
241         patched->f_op.splice_read = NULL;
242
243         atomic_set(&patched->ref_cnt, 0);
244         patched->unpatched = unpatched;
245 }
246
247 /**
248  * Default mcast_group_operations used in might_hook_fd()
249  *
250  * See documentation of koh_abort() and mcast_delete_group() for further info.
251  */
252 static struct mcast_group_operations __mcast_group_ops = { .no_subscribers = koh_abort,
253                                                             .no_publishers =
                                                                 mcast_delete_group };
254
255 /**
256  * might_hook_fd - @fd might be hooked before syscall return
257  * @fd: file descriptor
258  * @filename: sys_{open,openat,creat} argument
259  * @flags: ditto
260  * @mode: ditto
261  *
262  * A call to this function might hook @fd for the current process if
263  * 'thischrdev_getsubscribers_lock' returns any subscriber.
264  */
265 static void might_hook_fd(long fd, const char __user *filename, int flags,
266                           umode_t mode) {
267         struct file *filp = IS_ERR_VALUE(fd) ? NULL : get_filp(fd);
```

```
268         LIST_HEAD(subscribers);
269         char *__filename = (char *)get_zeroed_page(GFP_KERNEL);
270         int __filename_len;
271
272         __filename_len = strncpy_from_user(__filename, filename, PAGE_SIZE);
273
274         thischrdev_getsubscribers_lock(filp, __filename, &subscribers);
275         if (!list_empty(&subscribers)) {
276                 /* hook file object if syscall didn't return error; else
277                  * free unused subscribers list */
278
279                 if (filp) {
280                         struct mcast_group *grp = mcast_create_group(&subscribers,
281                                                         MCAST_GROUP_SHARED, &
                                                             __mcast_group_ops);
282                         koh_hook(filp, filp, &filp->f_op, __patch_fop, grp);
283
284                         /* ENTREGA DE MENSAJES OMITIDA PARA FACILITAR LECTURA */
285                 }
286         }
287         thischrdev_getsubscribers_unlock();
288
289         free_page((unsigned long)__filename);
290 }
291
292 /*
293  *  hooks for 'sys_call_table' system calls
294  */
295 asmlinkage long ____sys_open(const char __user *filename, int flags, umode_t mode) {
296         long ret = __orig_sys_open(filename, flags, mode);
297
298         might_hook_fd(ret, filename, flags, mode);
299
300         asmlinkage_protect(3, ret, filename, flags, mode);
301         return ret;
302 }
303
304 asmlinkage long ____sys_openat(int dfd, const char __user *filename, int flags, umode_t
        mode) {
305         long ret = __orig_sys_openat(dfd, filename, flags, mode);
306
307         might_hook_fd(ret, filename, flags, mode);
308
309         asmlinkage_protect(4, ret, dfd, filename, flags, mode);
310         return ret;
311 }
312
313 asmlinkage long ____sys_creat(const char __user *pathname, umode_t mode) {
314         long ret = __orig_sys_creat(pathname, mode);
315
316         might_hook_fd(ret, pathname, O_CREAT|O_WRONLY|O_TRUNC, mode);
317
318         asmlinkage_protect(2, ret, pathname, mode);
319         return ret;
320 }
321
322 /*
323  *  hooks for 'ia32_sys_call_table' system calls
324  */
325 #if defined(CONFIG_X86) && defined(CONFIG_COMPAT)
326 #include <linux/compat.h>
327
328 asmlinkage long ____compat_sys_open(const char __user *filename,
329                                     int flags, umode_t mode) {
330         long ret = __orig_compat_sys_open(__SC_DELOUSE(const char __user *, filename),
              __SC_DELOUSE(int, flags), __SC_DELOUSE(umode_t, mode));
331
```

```
332         might_hook_fd(ret, filename, flags, mode);
333
334         return ret;
335 }
336
337 asmlinkage long ____compat_sys_openat(int dfd, const char __user *filename, int flags,
338                                       umode_t mode) {
339         long ret = __orig_compat_sys_openat(__SC_DELOUSE(int, dfd), __SC_DELOUSE(const
                char __user *, filename), __SC_DELOUSE(int, flags), __SC_DELOUSE(umode_t,
                mode));
340
341         might_hook_fd(ret, filename, flags, mode);
342
343         return ret;
344 }
345 #endif
```

Apéndice E

Glosario

ABI interfaz a nivel de código máquina; entre otros, especifica cómo se invocará una función y cómo se pasan los argumentos.

ALSA Advanced Linux Sound Architecture (parte del kernel Linux que reemplaza a OSS) provee una API para controladores de dispositivo de tarjetas de audio.

buffer región de memoria usada temporalmente para almacenar datos que están siendo copiados.

chroot es una llamada a sistema que cambia el directorio raiz para un proceso y sus hijos, de modo que no pueden normalmente acceder ficheros fuera de éste. El entorno modificado se refiere como jaula chroot.

comm COMMand name, nombre del ejecutable excluyendo path; miembro de `struct task_struct`, ver `include/linux/sched.h`.

Dispositivo de caracteres provee acceso directo a un controlador de dispositivo, que puede ser accedido como una secuencia de bytes, e.g. puertos serie (`/dev/ttyS0`).

DOT lenguaje para la descripción de grafos en texto plano usado en el proyecto graphviz.

endianess se refiere a la interpretación de los bytes de una palabra. Un sistema *big-endian* almacena el byte más significativo de una palabra en la dirección más baja; por el contrario, en un sistema *little-endian* la dirección más baja estará ocupada por el byte menos significativo.

Ftrace Si el kernel se compila con esta característica, se añade una secuencia NOP de 5 bytes al principio de cada función del kernel, que es parcheada dinamicamente a una instrucción CALL.

FUSE Filesystem in UserSpacE permite implementar un sistema de ficheros como un proceso no privilegiado; el módulo *fuse* provee un puente a la interfaz del kernel.

getopt función de la librería C de GNU para parseo de opciones de linea de comando.

GUI Graphical User Interface, es una interfaz de usuario en la que éste interactúa con elementos gráficos haciendo uso de un ratón.

HTTP el HyperText Transfer Protocol es un protocolo de aplicación para sistemas hypermedia descrito en el RFC 2616.

E/S block I/O, i.e. lectura/escritura de un dispositivo de bloques (disco).

ioctl Input/Output ConTroL es una llamada a sistema para operaciones específicas de controlador de dispositivo.

JSON JavaScript Object Notation es usado para la transmisión de pares atributo–valor en texto plano.

overhead incremento en el tiempo de CPU comparado con una ejecución normal.

PID el Process ID es un número usado por el kernel para identificar un proceso o proceso ligero (hilo).

POSIX Portable Operating System Interface uniX, estándar de IEEE y The Open Group para mantener la compatibilidad entre sistemas UNIX y permitir la portabilidad de aplicaciones a nivel de código fuente.

procfs en los sistemas tipo UNIX, es un sistema de ficheros que exporta información de los procesos y otra información del sistema (típicamente montado en `/proc`).

root en los sistemas tipo UNIX, es el usuario que tiene todos los privilegios (UID=0).

stdout uno de los tres canales de comunicación abiertos cuando un programa entra en ejecución (fd=1), siendo los otros stdin (fd=0) y stderr (fd=2).

stream secuencia de bytes provista a lo largo del tiempo.

System.map es una tabla de símbolos que mapea cada símbolo del kernel a su dirección en memoria. Habitualmente puede encontrarse en el directorio `/boot`.

TGID Thread Group ID es el PID del primer hilo en un proceso multihilo.

TSR Terminate and Stay Resident es una llamada a sistema en DOS que permite terminar un programa y dejarlo residente en memoria hasta que éste es reactivado por una interrupción hardware o software.

udev reemplaza a *devfs*; gestiona nodos de dispostivo en `/dev`.

UID en sistemas tipo UNIX, el User ID es un número usado por el kernel para identificar un usuario.

valgrind herramienta para la depuración, detección de *leaks* de memoria y perfilaje.

Licencias

E.1. The GNU General Public License

Version 2, June 1991

59 Temple Place - Suite 330, Boston, MA 02111-1307, USA

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect

making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

Terms and Conditions For Copying, Distribution and Modification

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

   Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

   a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

   b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

   c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

   a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

   The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

   If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute

the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

   If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

   It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

   This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

No Warranty

11. Because the program is licensed free of charge, there is no warranty for the program, to the extent permitted by applicable law. Except when otherwise stated in writing the copyright holders and/or other parties provide the program "as is" without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The entire risk as to the quality and performance of the program is with you. Should the program prove defective, you assume the cost of all necessary servicing, repair or correction.

12. In no event unless required by applicable law or agreed to in writing will any copyright holder, or any other party who may modify and/or redistribute the program as permitted above, be liable to you for damages, including any general, special, incidental or consequential damages arising out of the use or inability to use the program (including but not limited to loss of data or data being rendered inaccurate or losses sustained by you or third parties or a failure of the program to operate with any other programs), even if such holder or other party has been advised of the possibility of such damages.

End of Terms and Conditions

Appendix: How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

one line to give the program's name and a brief idea of what it does.
Copyright (C) yyyy name of author

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) yyyy name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.
This is free software, and you are welcome to redistribute it under certain conditions; type 'show c' for details.

The hypothetical commands `show w` and `show c` should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w` and `show c`; they could even be mouse-clicks or menu items—whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program
'Gnomovision' (which makes passes at compilers) written by James Hacker.

signature of Ty Coon, 1 April 1989
Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

E.2. GNU Free Documentation License

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document "freeïn the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of çopyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The **"Document"**, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as **ƶou"**. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A **"Modified Version"** of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A **"Secondary Section"** is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The **Ïnvariant Sections"** are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to

be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The **Çover Texts"** are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A **"Transparent"** copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparentïs called **.Opaque"**.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The **"Title Page"** means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page"means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section **.Entitled XYZ"** means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as **.Acknowledgements"**, **"Dedications"**, **.Endorsements"**, or **"History"**.) To **"Preserve the Title"** of such a section when you modify the Document means that it remains a section .Entitled XYZ.according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or non-commercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "Historyïn the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History"section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled .Acknowledgements.or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled .Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled .Endorsements.or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled .Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties–for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the

same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "Historyïn the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled .ªcknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled .ºndorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an .ªggregateïf the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled .Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License .or any later version.applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright ©YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts."line with this:

with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Bibliografía

[AR05] Jonathan Corbet Alessandro Rubini. *Linux Device Drivers*. 3rd edition. O'Reilly, 2005.

[Auta] *Autoconf documentation*. 2012. URL: http://www.gnu.org/software/autoconf/manual/autoconf.html.

[Autb] *Automake manual*. URL: http://www.gnu.org/software/automake/manual/automake.html.

[Cpp] *C++ reference —C++98, C++03, C++11, C++14*. URL: http://en.cppreference.com/.

[DPB06] Marco Cesati Daniel P. Bovet. *Understanding the Linux kernel*. 3rd edition. O'Reilly, 2006.

[Fus] *FUSE API documentation*. URL: http://fuse.sourceforge.net/doxygen/index.html.

[Gno] *GTK+ 3 Reference Manual*. 2014. URL: https://developer.gnome.org/gtk3/stable/index.html.

[JD06] Jennifer Carroll Jeff Doyle. Second Edition. Vol. Volume I. CCIE Professional Development. Cisco Press, 2006.

[JK03] Abramo Bagnara Jaroslav Kysela. *Advanced Linux Sound Architecture - ALSA - Driver*. include/uapi/sound/asound.h, 2003.

[Kon07] Joseph Kong. *Designing BSD rootkits. An introduction to kernel hacking*. No Starch Press, 2007.

[Lina] *RCU on Uniprocessor Systems*. URL: http://www.kernel.org/doc/Documentation/RCU/UP.txt.

[Linb] *Using RCU to Protect Read-Mostly Linked Lists*. URL: http://www.kernel.org/doc/Documentation/RCU/listRCU.txt.

[McK03] Paul E. McKenney. *Kernel Korner - Using RCU in the Linux 2.5 Kernel*. 2003. URL: http://www.linuxjournal.com/article/6993.

[Ore] *Fun with Linux Kernel Modules*. 2010. URL: http://commons.oreilly.com/wiki/index.php/Network_Security_Tools/Modifying_and_Hacking_Security_Tools/Fun_with_Linux_Kernel_Modules.

[Vla11] Denys Vlasenko. *Ptrace documentation, draft #6*. 2011. URL: http://lwn.net/Articles/446593/.

[pou06] pouik. *Obtain sys_call_table on amd64(x86_64)*. 2006. URL: https://www.exploit-db.com/papers/13146/.

[pro15] Linux man-pages project. *Linux Programmer's Manual —PTRACE(2)*. 2015.

[sd01] devik sd. *Linux on the-fly kernel patching without LKM*. 2001. URL: `http://phrack.org/issues/58/7.html`.